MAI 1992

NUMERO 76

Le numéro 40 F

A PROPOS DU CLUB

J. Belin	Editorial	*1*
J. Belin	Présentation du Club Haute Performance	*2*
	Courrier du coeur	*2*

HP-28

J.L. Attenoux	Gestion d'erreurs	*4*
J.L. Attenoux	Mot de passe	*5*

HP-48

A. Aoufi	Nombres de Bernoulli	*8*
A. Aoufi	Intégrales Eulériennes complexes	*9*
A. Aoufi	Fonction d'Erreur	*9*

HP-71

A. Rottman	Désassembleur en Basic	*12*
E. Gengoux & A. Rottman	Problèmes de disquettes LIF	*13*

HP-95

J. Belin	Le nouveau HP-95	*18*
J. Belin	Programmes compatibles HP-95	*18*
J. Belin	Détection logicielle du HP-95	*19*

MS-DOS

J. Belin	Passerelle LIF-IBM (acte II)	*22*

EDITORIAL

Chers Amis,

Voici donc le nouveau *JPC*, mis en page par les membres du nouveau bureau. Nous l'avons voulu représentatif des différents domaines qui seront traités dans les prochains numéros. Nous espérons qu'il répondra à vos attentes.

Comme vous avez pu le voir dans le sommaire, nous étrennons une nouvelle rubrique consacrée au HP-95. Certains articles qui y sont consacrés ce mois-ci montrent que si nous sommes en présence d'un "Compatible IBM", celui ci possède des possibilités qui sont encore à découvrir, comme cela l'a été par le passé pour toutes les autre machines de la marque. Il est donc inutile de vous dire que la chasse est ouverte, et que si vous trouviez une "fonction cachée" ou tout autre indice sur l'organisation interne, non dévoilée par HP, nous sommes bien sûr prêts à en faire écho...

Comme vous pouvez le voir, nous serons de plus en plus amenés à utiliser d'autres machines, comme des IBM PC, pour faciliter le développement d'applications tournant sur nos machines de poche. Afin de ne pas déséquilibrer le contenu du journal avec des sources en assembleur ou en langage évolué (C, Pascal...) pouvant être beaucoup plus longs que ceux que nous publiions auparavant, nous avons décidé de modifier la présentation des listings. Vous pourrez en juger en fin de ce numéro. Nous attendons votre avis sur cette nouvelle présentation.

Parlons maintenant du HP-48. Depuis quelque temps, nous avons en notre possession ce que nous appelons les *IDS HP-48*, bien que cela soit sensiblement différent avec ce qui a été fait pour le HP-71. Ce document est accompagné d'un ensemble d'utilitaires permettant de développer des applications HP-48 sur IBM-PC. Nous vous informerons dans le prochain numéro de la méthode choisie pour vous distribuer ces éléments.

Pour finir, vous avez pu remarquer en lisant le sommaire, que ce journal compte relativement peu d'articles, écrits principalement par des membres du bureau. Il est donc relativement fin, mais nous espérons pouvoir, dès le prochain numéro, reprendre notre pagination normale, qui est de 36 à 40 pages. Nous n'osons plus vous rappeler, une fois de plus, qu'il est nécessaire que vous nous envoyez le plus vite possible des articles. En effet, il serait dommage que JPC disparaisse une fois de plus, mais cette fois pour cause d'articles promis mais non envoyés !

Ceci dit, nous vous souhaitons une bonne lecture...

Jacques Belin (123)

HAUTE PERFORMANCE

C'est le nom du club créé par la bien connue *Fédération Française des Jeux Mathématiques*, et dont le lancement est supporté par Hewlett Packard. Ses activités sont principalement centrées sur la HP-48, ainsi qu'une section dédiée au HP-95.

Leur journal, faisant huit à douze pages et paraissant tous les trois mois, se veut plus généraliste et moins "pointu" que *JPC*. Vous y trouverez donc des informations sur les nouveaux produits, des articles courts et quelques trucs et astuces concernant les calculateurs de poche.

Un de ses buts à long terme est de devenir le *club des clubs*, fédérateur des différents groupes d'utilisateurs (de HP-48 notamment) étant apparus en France ces dernières années.

Au lieu de nous livrer à une guerre fratricide, nous avons décidé de vivre en commun et de partager nos informations, afin de rendre nos deux clubs complémentaires. Il sera donc possible de voir dans leur journal de courts articles signés de membres de PPC, qui seront traités de façon plus fouillée dans notre revue.

Voici son adresse :

Club Haute Performance
Centre Châteaugaillard
1, Avenue Foch
94700 MAISON ALFORT

Le coût de l'adhésion est de 48 F par an (pour quatre journaux) et donne droit à la publication gratuite d'une petite annonce.

Jacques BELIN (123)

COURRIER DU COEUR

HOCLET Michel
9 allée du bois de la Cyprenne
91400 Orsay.
Tel pers. : (1) 69 07 32 41.
Tel prof. : (1) 69 08 59 86.

Vend :

Unité de disquettes HP 9114B, 2500FF; Imprimante ThinkJet, 1500FF; Interface video 80 col. HP 92198 et moniteur monochrome HP 92913A, 1500FF; Interface HP-IL/RS 232C HP 82164A, 1000FF.

VIRTUANI Fabrizio
via Valsesia 86
20152 Milano. Italia
Tel : 02/4566639.

Vend :

HP-71B version 2CDCC; HP-IL; RAM 96 Ko CMT; HP-IL card pour IBM-PC; Software HP-IL LINK pour IBM-PC; Internal Design Specification 3 vol.; Hardware Design Specification; HP-IL Design Specification; Interface RS-232C; Chargeur de batteries; 2 cables HP-IL de 1 m; Interface parallèle HP 82166A GP-IO; Modules : Math, Forth/Assembler, HP 41/71 Translator, Text Editor, Data Communication, AMPI Statistics, Finance, Circuit Analysis, Data Acquisition, Curve Fitting. Le tout pour 15000 FF avec documentation et emballages.

TORCHET Jean-Yves
Tel : 24 57 04 45 (province) après 20H

Vend :

2 HP-48SX, état neuf avec manuels, 1500FF chaque; extension mémoire 128K, 500FF; cable connexion PC, 500FF.

AOUFI Asdin
Tel. prof. : (1) 49 40 34 34

Vend :

HP-75C, parfait état avec documentation et chargeur, 3000FF.

HP-28

J.L. Attenoux	Gestion d'erreurs	*4*
J.L. Attenoux	Mot de passe	*5*

GESTION D'ERREURS

Ce programme permet de gérer les erreurs dans vos programmes pour HP-28C/S.

Utilisation :

```
#<N°Erreur> ERREND
```

Effet :

Affiche le message d'erreur correspondant, émet un BEEP d'erreur et arrête l'exécution du programme en cours. Ce programme vous permettra tout en simplifiant l'écriture de vos programmes de gagner quelques octets et de donner à vos applications un "Look HP" en vous servant des messages, et ils sont nombreux, qui existent dans la ROM de votre machine.

Listing :

```
Code asm   Mnémoniques       Commentaires

69C20      CON(5) 02C96      Début d'objet routine
LM
      début:
xxxxx      CON(5) fin-début  Longueur du programme
                              en quartets.
                               1BB et 1CC:00030.
                               2BB:0003C.
8F xxxxx   GOSBVL SAV.REG    Sauvegarde registres
                              D0,D1,B et D
                               1BB et 1CC:04EE2.
                               2BB:05081.
1B F510C   D0=(5) C015F      Pour ver 2BB seulement:
D2         C=0    A          Mise à zéro du
                              CMD NUMBER, pour ne
144        DAT0=C A           pas faire référence à
                              une instruction
                              particulière dans le
                              message d'erreur.
143        A=DAT1 A          A=Adr objet sur la pile
1131       D1=A              On va chercher à
179        D1=D1+10          récupérer l'info utile
143        A=DAT1 A          A=Num. erreur à traiter
8F xxxxx   GOSBVL LOAD.REG   Rappel registres sauvés
                               1BB et 1CC:04F19.
                               2BB:050B8.
8D xxxxx   GOVLNG ERROR      Exécution routine HP
                              en ROM.
                               1BB et 1CC:03358.
                               2BB:0396A.
142        A=DAT0 A          Fin de la routine
                              (standard).
164        D0=D0+5           808C   PC=(A)
      fin:
```

Avertissement :

Il convient d'être très prudent lors de la saisie de la chaîne à assembler, car une erreur de frappe risquerait de provoquer un "Memory Lost" de votre machine.

Pour une HP-28S assembler la chaîne suivante :

```
"69C20C30008F180501BF510CD2144143131179143 8F8B05
080A6930142164808C"
```

Puis stocker dans le répertoire HOME le System Object obtenu sous le nom 'ERREND'.

Pour une HP-28C version 1BB ou 1CC la chaîne est la suivante :

```
"69C20030008F2EE401431311791438F91F408D853301421
64808C"
```

Puis sauvegarder le System Object obtenu sous le nom 'ERREND'.

Exemples d'utilisation :

```
#5h ERREND        donne le message "Memory Lost"
#106h ERREND      donne le message "Syntax Error"
#204h ERREND      donne le message "Undefined Name"
#0 ou réel
   quelconque     n'émet pas de BEEP
```

Un exemple un petit peu plus concret pour une HP-28S :

Pour éviter de détruire complètement un répertoire utilisateur en exécutant l'instruction CLUSR par mégarde, il vous suffit de sauvegarder le petit programme suivant dans le répertoire principal HOME sous le nom 'CLUSR'. Voici la liste des actions à effectuer :

```
[HOME]
« #103h  ERREND  »
"21D205034C4553525"
Assembler cette chaîne résultat: 'CLUSR'
[STO]
```

Si vous exécutez maintenant la commande CLUSR le message "Improper User Function" apparaîtra et votre répertoire ne sera pas effacé. Pour annuler la redéfinition de CLUSR il suffit de détruire le programme CLUSR grâce à la commande PURGE.

Remarque :

Si aucun argument n'est présent sur la pile, si l'argument est invalide ou s'il n'y a pas de message d'erreur correspondant au numéro spécifié, cette routine émet un BEEP et arrête l'exécution du programme.

Conclusion :

J'ai tenu à expliciter au maximum le listing du programme car je pense que cela pourra donner des idées à certains pour se mettre à la programmation assembleur sur HP-28. Je conseille d'ailleurs à tous ceux qui sont désireux d'apprendre, de se référer à la bible en la matière, je veux bien sûr parler du formidable bouquin écrit par Paul COURBIS et Sébastien LALANDE, "Voyage au centre de la HP-28C/S", bouquin qui a permis à l'humble débutant que je suis d'écrire ce petit programme.

Jean-Louis ATTENOUX (83)

PASSWORD MACHINE

Ce programme pour HP-28S 2BB permet de protéger le contenu de sa machine par un mot de passe. Ce mot de passe est en fait une combinaison de touches à appuyer simultanément. Une fois le programme démarré il est alors impossible de l'arreter à moins d'effectuer la bonne séquence de touches ou un Memory Lost. Dans tous les cas vos données confidentielles seront protégées

Liste des routines :

- OFF — Permet d'éteindre la machine
- LOCK — Permet d'empêcher les tests et arrêt softs de la machine par les séquences de touches habituelles
- WUO — Wait Until Owner routine de lecture de la séquence de touches
- M\A — Programme liant l'ensemble des routines précédentes.

Listings :

Routine OFF :

```
"76C2085E8109F20" ASM 'OFF' STO
```

Routine LOCK :

```
69C20        CON(5) 02C96
34000        CON(5) 00043    Size=00043 Nibbles
8F18050      GOSBVL SAV.REG
1F4100C      D1(5)= C0014
3110         LC(2)  01
14B          A=DAT1 B
AE8          B=A    B
BEC          A=-A-1 B
0E66         A=A&C  B
BEE          C=-C-1 B
0E65         C=C&B  B
0E6E         A=A!C  B
149          DAT1=A B
8F8B050      GOSBVL LOAD.REG
142          A=DAT0 A
164          D0=D0+5
808C         PC=(A)

"69C20340008F180501F4100C311014BAE8BEC0E66BEE0E650
E61498F8B050142164808C"
ASM
'LOCK' STO
```

Remarque :
Cette routine effectue un OU Exclusif sur une zone de la mémoire de la HP-28S. Il faut donc l'effectuer deux fois pour retrouver l'état antérieur de sa machine.

Routine WUO :

```
69C20        CON(5) 02C96
05000        CON(5) 00050    Size=00050 Nibbles
8F18050      GOSBVL SAV.REG
1F5300C      D1(5)= C0035
30F          LC(1)  F
15D0         DAT1=C 1
3489200      LC(5)  00298    masque d'entrée
                              clavier
D5           B=C    A
3426000      LC(5)  00062    masque de sortie
                              clavier
8F1DA10      GOSBVL 01AD1
E9           C=C-B  A
8AE          ?C?0   A
DE           GOYES  -19
300          LC(1)  0
15D0         DAT1=C 1
8F8B050      GOSBVL LOAD.REG
142          A=DAT0 A
164          D0=D0+5
808C         PC=(A)
```

Qui est équivalente à :

```
"69C20050008F180501F5300C30F15D03489200D534260008F
1DA10E98AEDE30015D08F8B050142164808C"
ASM
'WUO' STO
```

La combinaison pour sortir est d'appuyer simultanément sur les touches [USER], [ENTER], [NEXT] et ['.'].

Routine M\A :

```
« CLLCD
" User password is ?"
1 DISP #0h ->RL LOCK    #0h ->RL est optionnel pour
                        une meilleure présentation
OFF LOCK WUO LOCK       Suivent les actions à
                        exécuter à chaque démarrage
SPEED CLOCK »           (SPEED et CLOCK sont des
                        exemples d' actions)
'M\A' STO
```

Avertissement :

Il convient d'être très prudent lors de la saisie de la chaîne à assembler, car une erreur de frappe risquerait de provoquer un "Memory Lost" de votre machine.

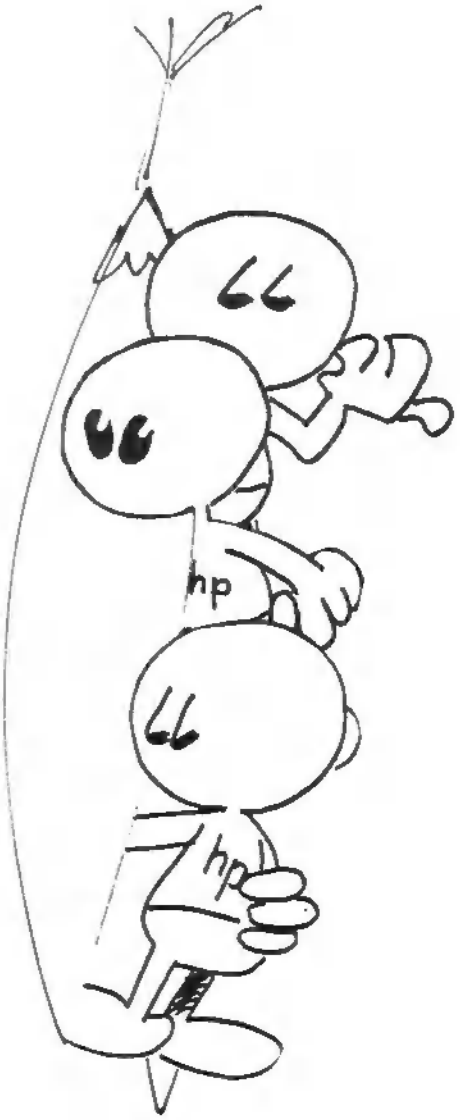
Enfin, je vous livre le moyen d'obtenir la version de votre machine par ce petit programme en assembleur qui comme la fonction OFF exécute la ROM de la HP-28S à partir d'une adresse donnée.

```
"76C20A000009F20"
ASM
'VER' STO
```

Voilà, j'espère que ces routines vous rendrons de nombreux services, pour ma part je les utilise quotidiennement.

Bonne Programmation à Tous !

Jean-Louis ATTENOUX (83)



HP-48

A. Aoufi	Nombres de Bernoulli	*8*
A. Aoufi	Intégrales Eulériennes complexes	*9*
A. Aoufi	Fonction d'Erreur	*9*

NOMBRES DE BERNOULLI

Jacob Bernoulli a découvert les nombres qui portent son nom au 17ème siècle. Ceux-ci interviennent dans de nombreux domaines de la Mathématique :
- Polynômes de Bernoulli,
- Formule sommatoire d'Euler-MacLaurin,
- Théorème de Kummer sur le problème de Fermat,
- Fonction ζ de Riemann.

Le programme proposé évalue ces nombres B_{2n} pour $1 \leq n \leq 13$ mais n'utilise pas la multiprécision.

Définition

La suite B_n vérifie la relation :

$$B_0 = 1,$$
$$B_1 = -1/2,$$
$$\Sigma B_k C^k_n = 0,$$
$$B_{2n+1} = 0.$$

Méthode

On évalue la suite B_{2n} qui vérifie :

$$B_0 = 1,$$
$$B_1 = -1/2,$$
$$\Sigma B_{2k} C^{2k}_{2n+1} = 0.$$

On définit des procédures de calcul exact sur des rationnels.

Le programme

Il se compose des modules suivants :

```
« → indice
  « { } indice 2 +
    + 2 + 0 CON → ber
    « 1 'ber(1,1)'
      STO  1 'ber(1,2)'
      STO -1 'ber(2,1)'
      STO  2 'ber(2,2)'
      STO 1 indice
      FOR p 1 2 p * - 2 1 p 1 -
        FOR l
          IF p 1 >
          THEN 2 p * 1 + 2 l * COMB
           'ber(l+2,1)' →NUM *
           'ber(l+2,2)' →NUM
           SIMPLIFIE SOMME
          END
        NEXT 2 p * 1 + * NEG SIMPLIFIE
        'ber(p+2,2)' STO 'ber(p+2,1)' STO
      NEXT
      ber
    »
  »
»
'BER' STO

« → n d
  « n d PGCD → pgcd
    « n pgcd / d pgcd / → nn dd
      «
        IF dd 0 <
        THEN nn NEG
         'nn' STO dd NEG
         'dd' STO
        END nn dd
      »
    »
  »
»
'SIMPLIFIE' STO

« → n1 d1 n2 d2
  « d1 d2 PPCM → ppcm
    « n1 ppcm * d1 / n2 ppcm * d2 / +
      ppcm SIMPLIFIE
    »
  »
»
'SOMME' STO

« → a b
  « a b * a b PGCD /
  »
»
'PPCM' STO

« → a b
  «
    IF b 0 ==
    THEN a
    ELSE b a b MOD
     PGCD
    END
  »
»
'PGCD' STO
```

Mode d'emploi

1) Entrer n inférieur ou égal à 13 et supérieur à 1 comme argument dans la pile.

Par exemple :
n : 5

2) Taper BER. On obtient le tableau des nombres de Bernoulli d'indice pair contenant les numérateurs et dénominateurs.

Noter que les 2 premiers éléments sont: B_0, B_1 puis B_2, ... B_{2n}.

Par exemple :

```
[[  1  1 ]
 [ -1  2 ]
 [  1  6 ]
 [ -1 30 ]
 [  1 42 ]
 [ -1 30 ]
 [  5 66 ]
```

Les numérateurs et dénominateurs des nombres de Bernoulli croissent très vite et changent alternativement de signe. Un recours à la multiprécision serait utile. Si le coeur vous en dit!.

Asdin AOUFI (562)

INTEGRALES EULERIENNES COMPLEXES

Le HP-48SX est un fantastique outil pour faire du calcul numérique mais présente le même défaut que son aïeul HP-28S sur la fonction Γ d'Euler. Il ne peut l'évaluer dans le plan complexe!

Nous proposons un programme de calcul qui utilisant les nombres de Bernoulli - cf article précédent - se base sur la formule de Binet pour :
- calculer Γ,
- calculer B.

Les programmes

```
« → z
  « 'LN(2*π)/2+(z-.5)*LN(z)-z' →NUM
    1 13
    FOR m
     'B(m+2,1)/(B(m+2,2)*2*m
     *(2*m-1)*z^(2*m-1))'
     →NUM +
    NEXT EXP
  »
»
'GAMMA' STO

« → x y
  '(GAMMA(x)+GAMMA(y))/GAMMA(x+y)'
  →NUM
»
'BETA' STO

13 BER 'B' STO
```

1) Entrer x comme argument dans la pile.
Par exemple :
x : 5

2) Taper GAMMA.On obtient la valeur de Γ
Par exemple :
23.9999999998

3) Entrer x et y comme arguments dans la pile.
Par exemple :
2 2

4) Taper BETA.On obtient la valeur de B
Par exemple :
0.33350419019 à comparer à 1/3.

5) Entrer x complexe comme argument dans la pile.
Par exemple :
x : (5,5)

6) Taper GAMMA.On obtient la valeur de Γ
Par exemple :
(-0.974395241698,2.00668988275)

Compte tenu du faible ensemble de nombres de Bernoulli considéré on obtient une précision moyenne .

Asdin AOUFI (562)

FONCTION D'ERREUR

Le numéro précédent *JPC 73/75* présentait divers programmes de Robert Pulluard pour évaluer la fonction d'erreur à l'aide de la série hypergéométrique.

Revenons-y aujourd'hui pour proposer une autre méthode de calcul qui tient compte des caractéristiques de la fonction UTPN.

L'astuce est lisible dans le programme ERFB.

Les programmes

```
«
  → x '2/√π*f(0,x,EXP(-T^2),T)' →NUM
»
'ERFA' STO
```

```
« → x
  « 1 0 .5 x UTPN 2 * -
  »
»
'ERFB' STO
```

Mode d'emploi :

1) Entrer ■ comme argument dans la pile.
Par exemple en mode STD:
x : 0.5

2) Taper ERFA. On obtient la valeur d'ERF(x);
.520499877811
IERR = 4.6E-12

3) Taper ERFB. On obtient instantanément la valeur approchée d'ERF(x);
.520499877814

On observe le même résultat à epsilon près entre les deux programmes avec un avantage en temps considérable pour le second.

Si vous faites des calculs de diffusion il vous sera sûrement très utile comme élément de base pour des formules complexes , pour ma part je l'utilise quotidiennement.

Asdin AOUFI (562)

HP-71

A. Rottman	Désassembleur en Basic	*12*
E. Gengoux & A. Rottman	Problèmes de disquettes LIF	*13*

L'OPERATION DE DESASSEMBLAGE

La présentation des instructions de l'ASSEMBLEUR trouve une extension très intéressante dans les procédures de désassemblage.
Un des programmes les plus intéressants de désassemblage a été publié par J. Taillandier dans *Micro Revue 5/1985* et sera discuté en détail dans cet article. Il faut souligner que cette discussion est à la fois utile pour l'apprentissage du BASIC et de l'assembleur.

Structure et façon de travailler du programme

1- Variables essentielles :

A0=Adresse; N$(I$)=Code; L$= Instruction. Le but du programme est d'écrire ces 3 variables (ligne 8520).

On a montré précédemment que l'ensemble des instructions de l'Assembleur sont codés avec des symbits ayant de 2 à 18 nibs. Certaines instructions contiennent dans le code une sélection du champs du registre, une distance de transfert ou une constante.

Les codes des instructions peuvent commencer par des 16 caractères hexadécimaux (0 à F) qui introduisent donc une division en classes (les lignes 300, 705, 1010, 1015, 1500, 1600, 1620, 1640, 5005, 5000, 5500, 8300, 8345, 8365, 8400 du programme). Les classes les plus développées sont les 1 et 8.

L'algorithme du programme est basé sur une analyse des 4 premiers caractères du code qui sont retenus dans la variable N$. On introduit une notation spéciale pour le 2ème, 3ème et 4ème caractères respectivement M$, O$ et P$.

En fonction de la valeur du premier caractère la ligne 294 branche vers une classe différente définie par ce premier caractère de l'instruction.

Si par exemple le premier caractère est 0 on est transféré à la ligne 300. Les instructions de cette classe ont 2 à 4 bits. Les instructions à 2 bits ont le deuxième caractère variable de 0 à F avec E qui manque, parce que les instructions classe 0 à 4 bits ont toutes E en deuxième position et le caractère qui fait la différence est le quatrième.

Alors la procédure adoptée est la suivante :
pour 2 ou 4 bits on lit d'abord dans un tableau A$( ) l'ensemble des instructions de la classe respective; le tableau A$( ) est dimensionné maximalement DIM A$(15)[6] et on choisit après, en fonction du caractère 2, (M$) le terme A$(HTD(M$)) et similairement pour les instructions 4 bits le terme A$(HTD(P$)), (P$ étant le 4ème caractère). Il faut ajouter que dans le code de l'instruction, la sélection du champs du registre se fait pour chaque instruction par addition des caractères appropriés.

Comme particularité du BASIC il faut souligner la mise en correspondance du tableau A$( ) avec un bloc DATA, en faisant préalablement RESTORE "# ligne". Si on examine la classe 1 on a des instructions à 3, 4, 6, 7 nibs.
Pour toutes les instructions, dont le deuxième caractère M$>5, l'instruction est construite dans le programme (à partir de la ligne 815), instructions classe 1, avec 6 comme deuxième caractère du code, et ayant la forme :

```
D0=D0+n
```

Le nombre de caractères lus, pour chaque instruction, tient compte de nombre de nibs de l'instruction respective.
La fonction FNP est équivalente à PEEK.
Les autres fonctions FNA$, FNB$, FNC$ servent pour calculer les adresses relatives nécessaires pour effectuer respectivement un transfert court, long et très long (256, 4096, 65536 octets).
En général la moitié de ces distances est positive et l'autre moitié (supérieure) est négative.

Maintenant on peut se demander si au lieu d'utiliser un fichier contenant l'ensemble des instructions de l'Assembleur, introduites dans le programme sous forme de blocs DATA, on ne peut extraire ces instructions, du module FORTH-ASSEMBLEUR. Or cette opération est possible, et elle est faite par le programme AS1.

Ce programme exploite la table qui se trouve à partir de l'adresse E74D7 et la structure de cette table est la suivante:
Classe de l'opcode 2 nibs, longueur du texte 1 nib, l'instruction 12 nibs, les drapeaux 2 nibs, la longueur du code de l'instruction 1 nib, le codage de l'instruction 5 nibs, la variable A 1 nib et la variable B 1 nib.

Les opcodes sont groupés en 43 groupes (0-42), et chaque groupe contient au plus 15 instructions.

Les drapeaux peuvent avoir 8 valeurs qui séparent les instructions fournissent une information pour la boucle principale ou précisent que l'instruction est la dernière de la chaîne.

L'arrangement des opcodes dans un groupe respecte une disposition ayant l'ordre alphabétique D-B-F-A-C-E-G. La fonction HASH (adresse EEF00) est

utilisée dans les procédures d'assemblage, pour déterminer le groupe.

L'algorithme consiste à prendre les trois bits d'ordre inférieur de chaque caractère de l'opcode et de former une chaîne connexe de 18 nibs. En prenant le mod 43 de cette valeur on obtient le groupe.

Aurel ROTTMAN (289)

PROBLEMES DES DISQUETTES

Pour pouvoir réaliser une liaison entre divers ordinateurs fabriqués par HEWLETT PACKARD, on a choisi le format appelé LIF (logical interface format). Ainsi par exemple pour le HP-75, on ■ adopté le format LIF1, et pour le HP-71, le format LIF; en même temps pour toutes les séries 200, 300, 500 on a adopté des procédures capables de lire et d'écrire des fichiers texte.

Ces procédures présentent une grande importance pour l'impression du bulletin de notre club. Mais il s'est avéré, qu'au moment où on ■ eu besoin d'effectuer cette opération, on ne savait pas quel était le type de la disquette utilisée, et en quoi elle différait, par rapport à une disquette standard, formatée sur le HP-71.

Le but de cette présentation est double :

1- permettre une compréhension du mécanisme de transfert HP71-IPC ou vice versa ,

2- donner la possibilité ■ auteurs d'articles, pour notre bulletin, et qui peuvent nous les envoyer sur une disquette, de formater cette disquette, pour qu'elle soit compatible à la fois avec le HP-71 et l'IPC.

Le formattage d'une disquette implique spécialement une écriture de l'enregistrement (secteur) zéro de la disquette, qui comporte 256 octets. (Voir *HP82401A HP-IL Interface Owner's Manuel for HP-71* p.233). Pour les disquettes HP-71 le catalogue des fichiers commence avec le secteur 2.
Si on analyse le secteur zéro de la disquette HP-71, et celui de la disquette LIF-IPC on a la situation reproduite dans le tableau suivant :

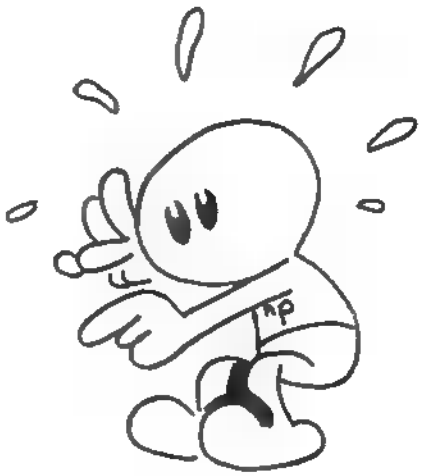
Octets Sect. zéro	Nature de l'enregis.	Néces sité HP71	Disquette HP71	Disquette LIF-IPC
1-2	Identific.	Oui	0800	0800
3-8	Etiquette	Oui	C49464020202	2554E4350202
9-12	Enreg.dép.	Oui	00000020	00000020
13-14	Sys.HP3000	Non	0100	0100
15-16	Non util.	Non	0000	0000
17-20	Long.Direc	Oui	000000D4	000000A1
21-22	Version	Oui	0010	0010
23-24	Non utilis	Non	0000	0000
25-28	Piste/surf	Oui*	000000D4	000000D4
29-32	No.surfac.	Oui*	00000020	00000020
33-36	Enreg/pist	Oui*	00000001	00000001
37-42	Date&temps	Non	092002227045	688052110000
43-256	Ext.&Maint	Non	zéro	zéro

*au cas ■ la valeur 'version' ■ 0

La conclusion, en examinant le tableau, est la suivante:

la disquette HP-71 peut être rendue compatible à l'IPC, par un changement des octets 19 et 20 du secteur zéro, les octets étant numérotés de 1 à 256. Il faut, en utilisant la fonction RREC$, du JPCROM, remplacer D4 par A1.

E. Gengoux(108) & A. Rottman(289)



Programme DISAS (Désassembleur, nécessite REVLEX)

```
  - DISAS J. TAILLANDIER MICRO 5/85
  9 OPTION BASE 0 ■ DESTROY J$
 10 DIM A$(15)[6],S$(7)[2],N$[4],M$[4],I$[20],C$(11)[6],J$[5]
 15 DIM A7$(7)[6],L$[30],L0$(15)[6],M0$(16)[5],M1$(16)[7]
 50 ■■■ S$(),C$(),A7$()
290 INPUT 'Start add? ';J$ ■ A=HTD(J$)
292 IF KEYDOWN THEN ■■■ ELSE N$=FNP$(A,4) @ A0=A ■ N=HTD(N$[1,1])+1
293 M$=N$[2,2] ■ O$=N$[3,3] ■ P$=N$[4,4] ■ A=A+2 ■ I$=N$[1,2]
294 ON N GOTO 310,707,1011,1020,1505,1605,1625,1645,5005,5000,5505,5525,8310,8360,8370,8410
  - CLASS 0
310 IF M$='E' THEN 320
312 RESTORE 9100 ■ ■■■ A$()
315 RESTORE 9100 ■ READ A$() @ L$=A$(HTD(M$)) @ GOTO 8520
320 RESTORE 9170 @ READ A$() @ L$=A$(HTD(P$))
325 IF O$='F' THEN L$=L$&' A' ELSE L$=L$&' '&S$(HTD(O$))
330 I$=I$&O$&P$ @ A=A+2 ■ GOTO 8520
  - CLASS 1
707 N=HTD(M$)+1
710 ■■ N GOTO 720,785,805,857,875,895,915,925,935,945,955,965,975,985,995,1005
720 RESTORE 9200 ■ READ A$() @ L$=A$(HTD(O$)) ■ A=A+1 ■ I$=I$&O$ ■ GOTO 8520
785 RESTORE 9210 @ READ A$() @ L$=A$(HTD(O$)) ■ A=A+1 @ I$=I$&O$ @ GOTO 8520
■■5 RESTORE 9220 @ READ A$() @ L$=A$(HTD(O$)) @ A=A+1 ■ I$=I$&O$ ■ GOTO 8520
857 RESTORE 9140 @ READ A$() @ L$=A$(HTD(O$)) @ A=A+1 ■ I$=I$&O$ ■ GOTO 8520
875 N=HTD(O$) @ L$=A7$(MOD(N,8))
880 IF N>7 THEN L$=L$&' B' ELSE L$=L$&' ■'
885 A=A+1 ■ I$=I$&O$ @ GOTO 8520
895 N=HTD(O$) @ A=A+2
900 IF N<8 THEN L$=A7$(N) @ L$=L$&' '&S$(HTD(P$))
905 IF N>7 THEN L$=A7$(MOD(N,8)) ■ L$=L$&' '&DTH$(HTD(P$)+1)
910 I$=I$&O$&P$ ■ GOTO 8520
915 N=HTD(O$) @ N=N+1 @ J$=STR$(N) @ I$=I$&O$
920 L$='D0=D0+ '&J$ @ A=A+1 ■ GOTO 8520
925 N=HTD(O$) ■ N=N+1 @ J$=STR$(N) @ I$=I$&O$
930 L$='D1=D1+ '&J$ ■ A=A+1 @ GOTO 8520
935 N=HTD(O$) @ N=N+1 @ J$=STR$(N) @ I$=I$&O$
940 L$='D0=D0- '&J$ @ A=A+1 @ GOTO 8520
945 A=A+2 ■ K$=O$&P$ @ I$=I$&K$
950 L$='D0=(2) '&REV$(K$) @ GOTO 8520
955 A=A+2 @ K$=O$&P$&FNP$(A,2) @ A=A+2 ■ I$=I$&K$
960 L$='D0=(4) '&REV$(K$) @ GOTO 8520
965 A=A+2 ■ K$=O$&P$&FNP$(A,3) @ A=A+3 ■ I$=I$&K$
970 L$='D0=(5) '&REV$(K$) @ GOTO 8520
975 A=A+1 @ I$=I$&O$ @ N=HTD(O$) ■ N=N+1 ■ J$=STR$(N)
980 L$='D1=D1- '&J$ @ GOTO 8520
985 A=A+2 @ K$=O$&P$ @ I$=I$&K$
990 L$='D1=(2) '&REV$(K$) @ GOTO 9520
995 A=A+2 @ K$=O$&P$&FNP$(A,2) ■ A=A+2 @ I$=I$&K$
1000 L$='D1=(4) '&REV$(K$) ■ GOTO 8520
1005 K$=O$&P$&FNP$(A,3) @ A=A+3 ■ I$=I$&K$ @ L$='D1=(5) '&REV$(K$) @ GOTO 8520
  - CLASS 2
1011 L$='P= '&M$ @ GOTO 8520
  - CLASS 3
1020 N=HTD(M$) ■ N=N+1 @ J$=STR$(N) @ K$=FNP$(A,N) ■ A=A+N @ L$='LC('&J$&') '
1030 I$=I$&K$
1035 L$=L$&REV$(K$) ■ GOTO 8520
```

```
  - CLASS 4
1505 K$=M$&O$ ■ I$=I$&O$
1507 IF K$='20' THEN L$='NOP3'
1510 IF K$='00' THEN L$='RTNC' ELSE L$='GOC    '&FNA$(K$,A-1)
1515 A=A+1 @ GOTO 8520
  - CLASS 5
1605 K$=M$&O$ @ I$=I$&O$
1610 IF K$='00' THEN L$='RTNNC' ELSE L$='GONC   '&FNA$(K$,A-1)
1615 A=A+1 @ GOTO 8520
  - CLASS 6
1625 K$=O$&P$ @ I$=I$&K$
1627 IF M$&K$='300' THEN L$='NOP4' @ A=A+2 @ GOTO 8250
1628 IF M$&K$='400' THEN L$='NOP5' ■ A=A+3 @ I$=I$&'0' @ GOTO 8520
1630 L$='GOTO    '&FNB$(M$&K$,A-1) ■ A=A+2 @ GOTO 8520
  - CLASS 7
1645 K$=M$&O$&P$ ■ I$=I$&O$&P$ @ A=A+2
1650 L$='GOSUB  '&FNB$(K$,A) @ GOTO 8520
5000 SFLAG 63 @ GOTO 5017 ! CLASS 9
5005 CFLAG 63 ! CLASS ■
5007 N=HTD(M$)+1
5010 ■ N GOTO 5065,5115,5150,5185,5220,5235,5240,5265,5280,5295,5017,5017,5305,5310,5325,5330
5017 IF FLAG(63) AND M$>'7' OR NOT FLAG(63) ■ M$='B' THEN RESTORE 9020 ELSE RESTORE 9000
5020 READ A$() ■ L$=A$(HTD(O$)) @ A=A+1 @ I$=I$&O$
5035 IF NOT FLAG(63) THEN L$=L$&' A' ELSE L$=L$&' '&S$(MOD(HTD(M$),8))
5040 GOTO 8500
5065 A=A+1 @ I$=I$&O$
5070 IF O$<>'8' ■ O$<'C' THEN L$=C$(HTD(O$)) ■ A=A+1 ■ I$=I$&O$ @ GOTO 8520
5075 IF O$='8' AND P$='F' THEN L$='INTOFF' @ A=A+1 @ I$=I$&P$ @ GOTO 8520
5080 IF ■$='8' AND P$='0' THEN L$='INTON'
5090 IF O$='E' THEN L$='SREQ?' @ GOTO 8520
5095 IF O$='C' THEN L$='C=P    '&P$ @ A=A+1 @ I$=I$&P$ ■ GOTO 8520
5100 IF O$='D' THEN L$='P=C    '&P$ @ A=A+1 @ I$=I$&P$ @ GOTO 8520
5105 IF O$='F' THEN L$='CPEX   '&P$ @ A=A+1 @ I$=I$&P$ @ GOTO 8520
5115 A=A+1 @ I$=I$&O$
5120 N=HTD(O$) ■ IF N<=7 THEN RESTORE 9060 ELSE 5125 @ READ A$() @ L$=A$(N)&'C'
5122 GOTO 8520
5125 N=POS('CDEF',O$)-1 @ L$='ASRBBSRBCSRBDSRB'[N*4+1,N*4+4]
5130 GOTO 8520
5150 A=A+1 @ I$=I$&O$
5155 IF O$='1' THEN L$='XM=0' @ GOTO 8520
5160 IF O$='2' THEN L$='SB=0' @ GOTO 8520
5165 IF O$='4' THEN L$='SR=0' @ GOTO 8520
5170 IF O$='8' THEN L$='MP=0' ■ GOTO 8520
5175 IF O$='F' THEN L$='CLAMST' @ GOTO 8520
5185 A=A+1 @ I$=I$&O$
5190 IF O$='1' THEN L$='?XM=0' @ GOTO 8500
5195 IF O$='2' THEN L$='?SB=0' @ GOTO ■
5200 IF O$='4' THEN L$='?SR=0' @ GOTO 8500
5205 IF O$='8' THEN L$='?MP=0' @ GOTO 8500
5220 L$='ST=0 '&O$ @ A=A+1 ■ I$=I$&O$ @ GOTO 8520
5235 L$='ST=1 '&O$ @ A=A+1 @ I$=I$&O$ @ GOTO 8520
5240 L$='?ST=0 '&O$ @ A=A+1 ■ I$=I$&O$ @ GOTO 8500
5265 L$='?ST=1 '&O$ @ A=A+1 @ I$=I$&O$ @ GOTO 8520
5280 L$='?P#0 '&O$ @ A=A+1 @ I$=I$&O$ @ GOTO 8500
5295 L$='?P=0 '&O$ ■ A=A+1 ■ I$=I$&O$ @ GOTO 8500
5305 K$=FNP$(A,4) ■ I$=I$&K$ @ L$='GOLONG '&FNC$(K$,A) ■ A=A+4 ■ GOTO 8520
5310 K$=FNP$(A,5) @ I$=I$&K$ ■ L$='GOVLNG '&XFN94004(K$) @ A=A+5 @ GOTO 8520
5325 K$=FNP$(A,4) @ I$=I$&K$ @ A=A+4 @ L$='GOSUBL '&FNC$(K$,A) @ GOTO 8520
```

```
5330 K$=FNP$(A,5) @ I$=I$&K$ ■ A=A+5
5335 L$='GOSBVL '&XFN94004(K$) ■ GOTO 8520
   - CLASS ■
5505 N=HTD(M$) @ IF N<8 THEN RESTORE 9080 ELSE RESTORE 9120
5510 READ A$() @ L$=A$(HTD(O$)) ■ A=A+1 ■ I$=I$&O$
5515 L$=L$&S$(MOD(N,8)) @ GOTO 8520
   - CLASS B
5525 N=HTD(M$) @ IF N<8 THEN RESTORE 9185 ELSE RESTORE 9060
5530 READ A$() ■ L$=A$(HTD(O$))
5535 L$=L$&' '&S$(MOD(N,8)) @ A=A+1 ■ I$=I$&O$ @ GOTO 8520
   - CLASS ■
8310 RESTORE 9080 @ READ A$() @ L$=A$(HTD(M$))&' A' ■ GOTO 8520
   - CLASS D
8360 RESTORE 9120 ■ READ A$() ■ L$=A$(HTD(M$))&' A' @ GOTO 8520
   - CLASS E
8370 RESTORE 9185 @ READ A$() @ L$=A$(HTD(M$))&' A' @ GOTO 8520
   - CLASS ■
8410 RESTORE 9060 @ READ A$() @ L$=A$(HTD(M$))&' A' @ GOTO 8520
   - SUBROUTINES
8500 GOSUB 8525 @ I$=FNP$(A,2) @ J=A ■ A=A+2 @ A0=A
8510 IF I$='00' THEN L$='RTNYES' ELSE L$='GOYES  '&FNA$(I$,J)
8520 PRINT DTH$(A0);TAB(8);I$;TAB(10+LEN(I$));L$ @ GOTO 292
8525 PRINT DTH$(A0);TAB(8);I$;TAB(10+LEN(I$));L$ @ RETURN
8600 DEF FNA$(I$,J)
8610   K=HTD(REV$(I$)) @ IF K>127 THEN K=K-256
8615   FNA$=DTH$(J+K) @ END DEF
8620 DEF FNB$(I$,J)
8625   K=HTD(REV$(I$)) ■ IF K>2047 THEN K=K-4096
8630   FNB$=DTH$(J+K) @ END DEF
8635 DEF FNC$(I$,J)
8640   K=HTD(REV$(I$)) ■ IF K>32767 THEN K=65536-K
8645   FNC$=DTH$(J+K) @ END DEF
8700 DEF FNP$(K,I)=PEEK$(DTH$(K),I)
   - DATA
8990 DATA ' P','WP','XS',' X',' S',' M',' B',' W'
8991 DATA OUT=CS,OUT=C,A=IN,C=IN,UNCFNG,CONFIG,C=ID,SHUTON,,C+P+1,RESET,BUSCC
8995 DATA DAT0=A,DAT1=A,A=DAT0,A=DAT1,DAT0=C,DAT1=C,C=DAT0,C=DAT1
9000 DATA ?A=B,?B=C,?A=C,?C=D,?A#B,?B#C,?A#C,?C#D,?A=0,?B=0,?C=0,?D=0,?A#0,?B#0,?C#0,?D&0
9020 DATA ?A>B,?B>C,?C>A,?D>C,?A<B,?B<C,?C<A,?D<C,?A>=B,?B>=C,?C>=A
9030 DATA ?D>=C,?A<=B,?B<=C,?C<=A,?D<=C
9060 DATA ASL,BSL,CSL,DXL,ASR,BSR,CSR,DSR,A=-A,B=-B,C=-C,D=-D,A=-A-1,B=-B-1,C=-C-1,D=-D-1
9080 DATA A=A+B,B=B+C,C=C+A,D=D+C,A=A+A,B=B+B,C=C+C,D=D+D,B=B+A,C=C+B
9090 DATA A=A+C,C=C+D,A=A-1,B=B-1,C=C-1,D=D-1
9100 DATA RTNSXM,RTN,RTNSC,RTNCC,SETHEX,SETDEC,RSTK=C,C=RSTK,CLRST
9110 DATA C=ST,ST=C,CSTEX,P=P+1,P=P-1,,RTI
9120 DATA A=0,B=0,C=0,D=0,A=B,B=C,C=A,D=C,B=A,C=B,A=C,C=D,ABEX,BCEX,ACEX,CDEX
9140 DATA D0=A,D1=A,AD0EX,AD1EX,D0=C,D1=C,CD0EX,CD1EX,D0=AS,D1=AS
9150 DATA AD0XS,AD1XS,D0=CS,D1=CS,CD0XS,CD1XS
9170 DATA A=A&B,B=B&C,C=C&D,D=D&C,B=B&A,C=C&B,A=A&C,C=C&D,A=A!B,B=B!C
9180 DATA C=C!A,D=D!C,B=B!A,C=C!B,A=A!C,C=C!D
9185 DATA A=A-B,B=B-C,C=C-A,D=D-C,A=A+1,B=B+1,C=C+1,D=D+1
9190 DATA B=B-A,C=C-B,A=A-C,C=C-D,A=B-A,B=C-B,C=A-C,D=C-D
9200 DATA R0=A,RR1=A,R2=A,R3=A,R4=A,,,,R0=C,R1=C,R2=C,R3=C,R4=C,,,
9210 DATA A=R0,A=R1,A=R2,A=R3,A=R4,,,,C=R0,C=R1,C=R2,C=R3,C=R4,,,
9220 DATA AR0EX,AR1EX,AR2EX,AR3EX,AR4EX,,,,CR0EX,CR1EX,CR2EX,CR3EX,CR4EX,,,
```

HP-95

J. Belin	Le nouveau HP-95	*18*
J. Belin	Programmes compatibles HP-95	*18*
J. Belin	Détection logicielle du HP-95	*19*

LE NOUVEAU HP-95

Annoncé depuis quelque temps déjà, la nouvelle version du HP-95 devrait être disponible au moment où vous lirez ces lignes.

La principale différence réside dans sa mémoire principale portée à 1 Mo, au lieu des 512 Ko de la première version. Ceci permettrait donc de porter la mémoire de base à la limite fatidique des 640 Ko de MS-DOS, et réserver les 384 Ko restants au disque dur C.

D'autre part, de nouvelles fonctions DOS auraient été ajoutées, ainsi qu'une meilleure gestion de la mémoire.

Le seul point noir réside dans son prix, environ 1000F plus cher que la version 512 Ko, qui reste cependant commercialisée.

Les personnes possédant l'ancien modèle peuvent cependant bénéficier d'une opération de reprise organisée par EduCALC. En effet, cette bien connue société californienne propose à toute personne envoyant l'ancien modèle, accompagné d'un règlement de 195$, d'envoyer en retour une version à 1 Mo. Ce type d'opération est étudiée par certains distributeurs parisiens, mais reste conditionnée par HP France. Si cela est possible, nous vous en informerons dans le prochain numéro.

Jacques Belin (123)

TO RUN OR NOT TO RUN

Bien que le HP-95 soit un "compatible IBM", nous avons constaté que certains programmes PC fonctionnaient mal, ou ne tournaient pas du tout sur le HP-95. Ceci est souvent causé par certains programmes fonctionnant à un trés bas niveau, pour attaquer par exemple l'affichage ou le disque C. D'autre part, certains programmes, bien qu'il n'aient pas été ecrits spécifiquement pour le HP-95, ont un affichage qui peut être parfaitement adapté pour la machine.

Cet rubrique, que vous retrouverez régulièrement dans les prochains *JPC*, leur est consacré. Nous vous indiquerons quelle est la version testée, car il est fort possible qu'une nouvelle version du programme fonctionne mieux que la précédente, ou inversement. Nous vous indiquerons aussi si ils sont dans le domaine public (*freeware*), en *shareware*, ou commerciaux. Il est bien entendu que si nous nous efforcerons de faire circuler ces deux premiers types de programmes, nous ne pourrons pas redistribuer des programmes commerciaux...

Programmes Conseillés :

FSD.COM : Ce programme, dont l'écran de présentation indique *IBM Internal Use Only*, est un débugger permettant l'analyse de programmes COM ou EXE. Sur IBM, l'écran affiche différentes zones : Registres CPU, Code désassemblé du programme et contenu de deux zones mémoire en Hexa-décimal (la deuxième comporte en outre la correspondance en ASCII). Sur l'affichage du HP-95, les deux premières zones (les plus importantes à mon avis), sont visibles en totalité.

Programmes tournant imparfaitement :

PC TOOLS (version 1.00, commercial) : Sur cet utilitaire bien connu, il est possible d'afficher l'état de fragmentation des disques (MAPPING). Or si cela est possible sur le disque A (carte mémoire), nous ne pouvons pas visualiser celui du disque C, car le programme ne trouve pas la FAT. Cela est peut-être dû au fait que ce disque ne comporterai pas de table de partition sur le HP-95.

POWER METER (The Database Group, version 1.5, commercial ?) : Ce programme, permettant d'effectuer tout une série de tests de performances du PC, fonctionne relativement bien, sauf en ce qui concerne certains accès disques, sûrement pour les mêmes raisons que PC TOOLS. Par exemple, il est impossible de connaitre le temps d'accès du disque C, mais on peut peut voir celui du disque A, qui est de 2 milli-secondes (avec un taux de transfert de plus de 2300 Ko/s) !

Programmes ne fonctionnant pas :

SIDEKICK (Borland, version 1.52, commercial) : Ensemble d'utilitaires résident. Plante le système.

SPEED (Landmark, version 0.99, Freeware) : Programme de test de performances du CPU. Plante (*Internal Stack Failure*).

CORETEST (Core, version 2.4, Freeware) : Programme de test de performances de disque dur. Plante (*Disk 0 not present*).

BRKBOX (Tim Burmeister, version 1.1, Freeware) : Programme résident affichant en permanence, [redacted] haut de l'écran, l'état des lignes (données et handshakes) d'une interface RS-232. Plante.

C'est tout pour aujourd'hui, si vous avez testé d'autres programmes pouvant entrer dans ces catégories, n'hésitez pas à nous le faire savoir, cette rubrique leur est ouverte !

Jacques Belin (123)

DETECTION DU HP-95 PAR UN PROGRAMME MS-DOS

le HP-95 comportant quelques différences notables par rapport à un IBM PC classique, comme par exemple la taille de l'affichage, il peut être nécessaire à un programme savoir sur quelle machine est en train de s'exécuter.

Dans ce but, les développeurs du HP-95 [redacted] eu la bonne idée de nous fournir une fonction, accessible de façon standard par un appel d'interruption.

Cette fonction est appelée par l'intermédiaire de l'interruption 15, fonction 4d, sous-fonction d4. Si elle est appelée sur un PC quelconque, les valeurs retournées dans les registres seront nulles, ou incohérentes, selon les différents BIOS.

Par contre si la fonction est appelée sur le HP-95, les valeurs retournées dans les registres du microprocesseur seront :

Registre AH = code ASCII du caractère 'H'
AL = code ASCII du caractère 'P'
CH = code de la machine (1 pour le HP-95)
CL = numéro de version de la machine.

En assembleur, nous pouvons écrire une fonction utilisant cette interruption comme ceci :

```
detect_hp95 :
        mov   ax,4dd4h      ;initialisation
        int   15            ;appel interruption
        test  ax,'HP'       ;AX = 'HP' ?
        jne   notfnd        ;non, on va plus bas
        test  ch,01         ;CH = 1 ?
        jne   notfnd        ;non, on va plus bas
        mov   ax,0001       ;c'est un HP-95
        jmp   fin           ;vers la fin
notfnd : sub  ax,ax         ;AX = 0
fin :   ret                 ;fin de la fonction
```

En cas de détection du HP-95, le registre AX contient la valeur 1, sinon on y trouvera la valeur 0.

Vous pouvez noter que le contenu du registre CL (version du HP-95) n'est pas pris en compte. Si les prochaines versions du HP-95 présentent des différences importantes, il faudra en tenir compte, peut-être dans la valeur retournée. Il est donc nécessaire, pour éviter tout problème futur, d'effectuer le test de la valeur retournée par `detect_hp95` sur la valeur "0" et non sur la valeur "1", si on désire seulement savoir si on tourne sur HP-95 ou sur PC.

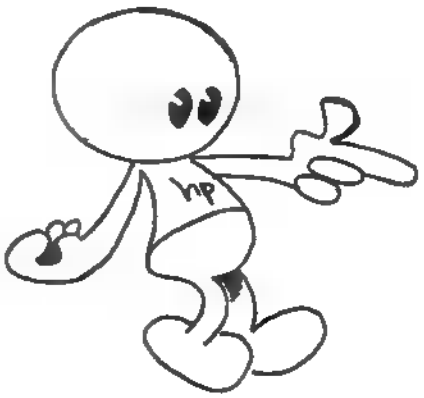
Dans le même esprit, voici la même routine écrite [redacted] langage C :

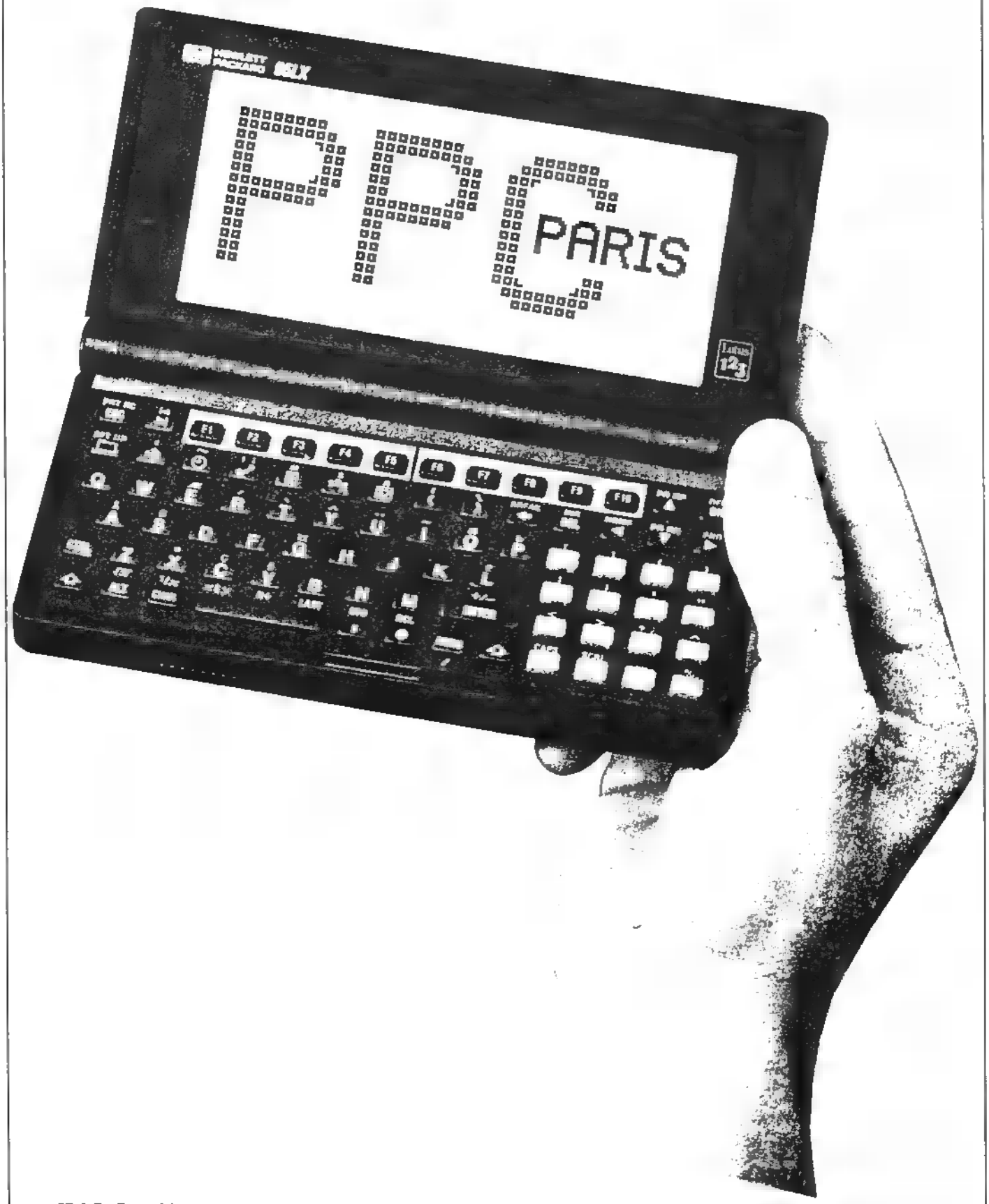
```
int detect_hp95(void)
  {
   union [redacted] regs;
   int retour=0;

   regs.x.ax= 0x4dd4;
   int86(0x15, &regs, &regs);
   if((regs.h.bh == 'H') && (regs.h.bl == 'P')
                          [redacted] (regs.h.ch == 0x01))
      retour=1;
   return(retour);
  }
```

En cas d'exécution sur HP-95, la routine retournera la valeur "1", sinon ce sera la valeur "0".

Jacques Belin (123)




HEWLETT PACKARD 95LX
JPC
PARIS
Lotus 1-2-3

MS-DOS

J. Belin Passerelle LIF-IBM (acte II) *22*

POUR UNE POIGNEE DE FONCTIONS

Récemment, je vous ai présenté une toolbox permettant de lire des disquettes LIF sur un IBM-PC. Comme promis, je vous présente aujourd'hui quelques commandes MS/DOS permettant de lister un catalogue et de manipuler certains fichiers.

LIFCAT

Cette commande permet d'afficher le catalogue d'une disquette LIF.

Syntaxe :

```
lifcat [-a text] [-c] [-d device] [-e] [-l] [-t TYPE]
```

Options :

`-a` : Permet d'afficher [redacted] texte après les entrées affichées, pour indiquer par exemple un nom de disquette dans le cas d'un tri de plusieurs disquettes dans un même fichier (methode utilisée pour imprimer la premiere partie du catalogue de la programathèque HP-71).

`-c` : Permet d'afficher le catalogue avec un format court, c'est à dire seulement les noms de fichiers sur plusieurs colonnes.

`-d` : Sélection du lecteur de disquette. 0 pour le lecteur A, 1 pour B. Par défaut, c'est le lecteur 0.

`-e` : Affichage des fichiers effacés.

`-l` : Affichage au format long, c'est à dire avec les dates et heure de modification des fichiers, [redacted] en-tête donnant le Label de la disquette et des indications de longueur en fin de catalogue.

`-t` : Affichage d'un type donné de fichier. Les noms reconnus sont :
Pour la HP-41 : PR-41, KE-41, ST-41, XM-41, CA-41, WA-41, ML-41 et BU-41.
Pour le HP-71 : BASIC, LEX, BIN, DATA, FORTH, DATA, BIN, KEY, ROM71, APPT, GRAPH, ADRS, OBJ71 et SYM71.
Pour le HP-75 : TEXT75, BAS75, LEX75, APPT75, IOR75, VISI75 et ROM75.
Fichiers communs à plusieurs machines : TEXT et SDATA.

Vous pouvez utiliser les filtres standards du DOS (`more`, `sort`, `>`, `>>`) pour rediriger les sorties vers l'imprimante ou un fichier.

Exemples :

Affichage des fichers LEX au format court, sur le lecteur A:.

```
lifcat -d0 -c -t LEX
```

Impression au format long de tous les fichiers, sur le lecteur B:, y compris les fichiers effacés.

```
lifcat -eld1 >prn
```

Création d'un fichier trié par nom de fichier de 3 disquettes (avec indication du [redacted] de disquette apres chaque entrée). Seuls les fichiers TEXT sont pris en compte.

-mettre la disquette CAT1 dans le lecteur

```
lifcat -a cat1 -tTEXT >text.lif
```

-mettre la disquette CAT2 dans le lecteur

```
lifcat -a cat2 -tTEXT >>text.lif
```

-mettre la disquette CAT3 dans le lecteur

```
lifcat -a cat3 -tTEXT >>text.lif
sort <text.lif >text.tri
```

Notes :
1- Les types de fichiers non reconnus sont affichés avec leur code numérique.
2- Les types de fichiers doivent être tapés en majuscules.
3- Les dates sont affichées dans le format déterminé par la variable COUNTRY de MS/DOS.

LIFCP

Cette commande permet la copie de fichiers TEXT. Elle effectue une conversion des caractères accentués en IBM ou Roman8. Il est également possible de copier des fichiers d'un type différent.

Syntaxe :
DOS vers LIF : `lifcp [-b] dos_file drive:LIF_FILE`
LIF vers DOS : `lifcp [-b] drive:LIF_FILE dos_file`

`drive` est le numéro du lecteur : 0 pour A:, 1 pour B:

Options :

`-b` : Permet de stocker un fichier LIF sur un fichier MS/DOS spécial sans tenir compte de son type, ni faire de conversion R8-IBM. Le fichier DOS contient : un code de reconnaissance au format long (4 octets ayant la valeur 123123, suivi de l'entrée de directory (copie physique sur 32 octets), suivi de la totalité des enregistrements du fichier LIF lui-même. Cette option doit être réutilisée pour effectuer la copie dans l'autre sens (DOS->LIF). Cette option ne peut pas être utilisée sur un fichier DOS non créé avec cette option.

Exemples :

```
lifcp c:\src\toto.txt 0:TITI
lifcp -b 1:DESAL desal.lex
```

Notes :

Pour les noms de fichiers LIF, les majuscules et minuscules sont significatifs.
Certains fichiers HP-41 ayant des noms comprenant des caractères spéciaux ne pouront pas être manipulés.

ACP

Cette commande permet de générer un fichier LEX à partir d'un fichier créé par l'assembleur AREUH de Pierre David et Jannick Taillandier.

Syntaxe :

```
acp dos_file [drive:[LIF_NAME]]
```

drive est le numéro du lecteur : 0 pour A:, 1 pour B:. Par défaut c'est le lecteur 0.

exemples :

```
acp lex 1:
acp titi.ao 1:TOTOLEX
acp c:\lex\desal.ao
```

Note :
Si LIF_NAME n'est pas spécifié, le nom du fichier sera celui indiqué dans le source du LEX.

Compilation des programmes

Avec le TURBO C : La commande LIFCP utilisant un buffer adressé dans le tas externe, doit être compilé avec un modèle de mémoire générant des pointeurs longs. Pour cela, sélectionnez par exemple le modèle COMPACT. Pour réduire la taille des programmes, vous pouvez convertir les commandes LIFCAT et ACP en fichiers .COM. Pour cela, vous devez les compiler avec le modèle TINY puis utiliser la commande exe2bin pour créer le fichier .COM. Vous ne pouvez pas convertir la commande LIFCP en format .COM

Avec le Microsoft C : tous les programmes doivent être compilés avec l'option /AC.

Autre Listing : getopt.c

Cette fonction permet l'analyse des options au format UNIX. Un listing équivalent est fourni avec le TURBO C, mais pas avec le Microsoft C (à ma connaissance).

Je vous rappelle que les sources de ces listings et les programmes compilés, sont disponibles une disquette (ainsi que d'autres commandes). Pour l'obtenir, envoyez une disquette vierge club, nous vous la retournerons rapidement.

Jacques Belin (123)

Note : Les articles et listings de la première partie de cet article, présentés dans le numéro 73/75, présentent de nombreuses erreurs, dues notamment au formatteur utilisé pour mise page. Etant donné la taille de cet article, n'en publierons pas la version corrigée. Par contre, les personnes qui en feront demande pouront en obtenir une copie corrigée.

```
/*****************************************************************************/
/*****************************************************************************/
/**                              LIFCAT.C                                   **/
/**                 Listage de catalogue de disquette LIF                   **/
/**                                                                         **/
/**    Création : 06/10/90                 Auteur : JB                      **/
/**    Derniere modification : 06/02/91                                     **/
/**       Correction du calcul de la longueur des fichiers OBJ71, SYM71     **/
/**                                                                         **/
/*****************************************************************************/
/*****************************************************************************/
              /* normalisation de country, à inclure avant "#include <dos.h>"
                           si utilisation de cette structure avec TURBO C++ */
#define COUNTRY country

#include "getopt.c"
#include "lif.c"

#if defined(__TURBOC__)        /* fonctions dépendant du type de compilateur */
#include "lif.tc"
#else
#include "lif.msc"
#endif

void main(int argc, char *argv[])
  {
   int  device;                          /* numéro du lecteur de disquette */
   int  eff;                       /* flag d'affichage des fichiers effacés */
   int  ptext;                       /* flag de présence de texte optionnel */
   int  stype;                                  /* flag de selection de type */
   char atxt[60], ttype[15];                /* paramètres des options -a et -t */
   int  formc, forml;         /* flags d'affichages de formats court et long */
   LIFDIR dir;                  /* structure contenant l'entrée du directory */
   char name[11];                                        /* nom de fichier LIF */
   int  private, secure;                   /* flags de protections du fichier */
   int  ntype;                            /* code numérique du type de fichier */
   int  long_file;                 /* longueur du fichier en enregistrements */
   int  file_start;                  /* enregistrement de départ du fichier */
   GDH  gdh;             /* structure contenant la date et l'heure du fichier */
   long imp1, imp2;                /* variables dépendant du type de fichier */
   char status;                     /* état du fichier ' ', 'S', 'P' ou 'E' */
   char ligne[100];                                        /* ligne affichée */
   char typef[10], datef[40];            /* type de fichier et date, ■ Texte */
   unsigned int  longd;          /* longueur en enregistrements du directory */
   unsigned int  nb_files=0;                  /* nombre de fichiers affichés */
   unsigned long long_bytes;                 /* lonqueur du fichier en octets */
   unsigned int  ltotale=0;          /* somme longueur des fichiers ■ enreg. */
   long ltotalo=0L;                  /* ■■■■ longueur des fichiers ■ octets */
   int  errflag;                                           /* flag d'erreur */
   struct country c_country;         /* struct contenant le format de la date */
   int i, j;
   char c;

   formc = 0;                     /* initialisation des paramètres par défaut */
   forml = 0;
   eff = 0;
   stype = 0;
   device = 1;
   ptext=0;
   strcpy(atxt, "");

                                                   /* analyse des parametres */
   errflag=0;
   SW='-';
   while((c=(char) getopt(argc, argv, "a:cd:elt:")) != EOF)
     {
      switch(c)
        {
         case 'a' : strcpy(atxt, optarg);                    /* texte ajouté */
                    ptext=1;
                    break;

         case 'c' : formc=1;                                  /* format court */
                    break;

         case 'd' : device = *optarg-'0';          /* lecteur de disquette */
                    if((device != 0) ■ (device != 1))
                       errflag++;
                    break;

         case 'e' : eff = 1;                             /* fichiers effacés */
                    break;

         case 'l' : forml = 1;                                /* format long */
                    break;

         case 't' : strcpy(ttype, optarg);                /* selection de type */
                    strupr(ttype);                     /* mise en forme du type */
                    for(i=strlen(ttype); i<6; i++)    /* avec espaces en fin */
                       strcat(ttype, " ");
                    stype=1;
                    break;

         case '?' : errflag++;                                  /* si erreur */
                    break;
        }
     }


   if(errflag)                                            /* fin si erreur */
     {
      fprintf(stderr,
     "usage : lifcat [-a text] [-c] [-d drive_number] [-e] [-l] [-t type]\n");
      exit(-1);
     }

   init_dsk();                       /* initialisation du lecteur de disquette */

                                       /* lecture de l'entête de la disquette */
   errflag=read_dsk(device, 0, buff_sect);
   if(errflag)                                            /* fin si erreur */
     {
      fprintf(stderr, "error : not a LIF disk");
      fin(-1);
     }

   if(forml)                                  /* si format long selectionné */
     {
      fprintf(stdout, "  DISK : %s\n", buff_sect+2);      /* affichage label */

      if(formc != 1)                                  /* si pas de format court */
        {                                   /* affichage d'un en tête de catalogue */
         fprintf(stdout, "NAME        S TYPE     SIZE ");
         fprintf(stdout, " DATE           ");
         if(ptext)                                     /* si presence de texte */
            fprintf(stdout, " TEXT");
         fprintf(stdout, "\n");
         fprintf(stdout, "==========================");
         fprintf(stdout, "================");
         if(ptext)
           {
            j=strlen(atxt);
```

```
            strcpy(name, "=");
            for(i=1; i<=j+1; i++)
               strcat(name, "=");
            fprintf(stdout, "%s", name);
          }
        fprintf(stdout, "\n");
      }
  }
                          /* longueur en enregistrements de la disquette */
longd = ((int) buff_sect[18]*256)+(int) buff_sect[19];

curr_sect=1;                /* initialisation avant lecture du directory */
curr_dir=7;
while(read_dir(device, &dir))          /* boucle de lecture du directory */
  {
                                              /* decomposition de l'entrée */
   break_dir(name, &private, &secure, &ntype, &file_start, &long_file,
                                          &gdh, &imp1, &imp2, &dir);
   if((ntype != 0) || (eff==1))    /* si fichier non effacé ou option -e */
     {
      switch(ntype)                                  /* selection du type */
        {
         case 0      : strcpy(typef, "dddddd");          /* fichier effacé */
                       long_bytes=long_file*256L; /* calcul longueur aff.*/
                       break;

         case TEXT   : strcpy(typef, "TEXT  ");
                       long_bytes=long_file*256L;
                       break;

         case SDATA  : strcpy(typef, "SDATA ");
                       long_bytes=imp1;
                       break;

         case DATA   : strcpy(typef, "DATA  ");
                       long_bytes=imp1*imp2;
                       break;

         case FORTH  : strcpy(typef, "FORTH ");
                       long_bytes=(long) imp1/2;
                       break;

         case BIN    : strcpy(typef, "BIN   ");
                       long_bytes=(imp1-11L)/2;
                       break;

         case LEX    : strcpy(typef, "LEX   ");
                       long_bytes=(imp1+1L)/2;
                       break;

         case KEY    : strcpy(typef, "KEY   ");
                       long_bytes=(imp1+1L)/2;
                       break;

         case BASIC  : strcpy(typef, "BASIC ");
                       long_bytes=(imp1-11L)/2;
                       break;

         case ROM71  : strcpy(typef, "ROM71 ");
                       long_bytes=(long) imp1/2;
                       break;

         [redacted] APPT   : strcpy(typef, "APPT  ");
                       long_bytes=(long) imp1/2;
                       break;

         case GRAPH  : strcpy(typef, "GRAPH ");
                       long_bytes=(imp1-10L)/2;
                       break;

         case ADRS   : strcpy(typef, "[redacted]  ");
                       long_bytes=(imp1-24L)/2;
                       break;

         [redacted] OBJ71  : strcpy(typef, "OBJ71 ");
                       long_bytes=(long) imp1/2;
                       break;

         case SYM71  : strcpy(typef, "SYM71 ");
                       long_bytes=(long) imp1/2;
                       break;

         case PR_41  : strcpy(typef, "PR-41 ");
                       long_bytes=imp1;
                       break;

         case CA_41  : strcpy(typef, "CA-41 ");
                       long_bytes=imp1;
                       break;

         case XM_41  : strcpy(typef, "XM-41 ");
                       long_bytes=imp1;
                       break;

         case ML_41  : strcpy(typef, "ML-41 ");
                       long_bytes=imp1;
                       break;

         [redacted] BU_41  : strcpy(typef, "BU-41 ");
                       long_bytes=imp1;
                       break;

         [redacted] ST_41  : strcpy(typef, "ST-41 ");
                       long_bytes=imp1;
                       break;

         case LEX75  : strcpy(typef, "LEX75 ");
                       long_bytes=long_file*256L;
                       break;

         [redacted] TEXT75 : strcpy(typef, "TEXT75");
                       long_bytes=long_file*256L;
                       break;

         case APPT75 : strcpy(typef, "APPT75");
                       long_bytes=long_file*256L;
                       break;

         [redacted] IOR75  : strcpy(typef, "IOR75 ");
                       long_bytes=long_file*256L;
                       break;

         case BAS75  : strcpy(typef, "BAS75 ");
                       long_bytes=long_file*256L;
                       break;

         case VISI75 : strcpy(typef, "VISI75");
                       long_bytes=long_file*256L;
                       break;

         case ROM75  : strcpy(typef, "ROM75 ");
```

```
                        long_bytes=long_file*256L;
                        break;

          default     : sprintf(typef, "%-6d", ntype); /*fich. non reconnu*/
                        long_bytes=long_file*256L;
                        break;
        }

      if(ntype)                                  /* si fichier ■ effacé */
        ltotale += long_file;          /* ajout du nombre d'enreg. utilisés */

                                 /* formatage date (suivant format système) */
      country(0, &c_country);
      if(c_country.co_date == 1)
        {
         sprintf(datef, " %02d/%02d/%02d %02d:%02d", gdh.day,
                                                     gdh.month,
                                                     gdh.year,
                                                     gdh.hour,
                                                     gdh.min);
        }
      else
        {
         sprintf(datef, " %02d/%02d/%02d %02d:%02d", gdh.month,
                                                     gdh.day,
                                                     gdh.year,
                                                     gdh.hour,
                                                     gdh.min);
        }


      status=' ';                            /* formatage état du fichier */
      if(private)
        status='P';
      if(secure)
        status='S';
      if(private && secure)
        status='E';

      j=strcmp(ttype, typef);
      if((stype==0) || (j==0))          /* si type de fichier selectionné */
        {
         nb_files++;
         ltotalo += long_bytes;          /* somme des longueurs ■ octets */
         if(formc)                                        /* si format court */
           {
            fprintf(stdout, "%10s ", name);           /* affichage du nom */
            if(!(nb_files % 7))   /* ligne suivante toutes les 7 entrées */
               fprintf(stdout, "\n");
           }
         else                                                    /* sinon */
           {                                  /* affichage nom+type+longueur */
            sprintf(ligne, "%s  %c %s %6ld ",
                                   name, status, typef, long_bytes);
            if(forml)                                        /* si format long */
               strcat(ligne, datef);
            strcat(ligne, "  ");       /* affichage texte, le cas echéant */
            strcat(ligne, atxt);
            fprintf(stdout, "%s\n", ligne);
           }
        }
    }
  }

if(formc && (nb_files ■ 7))   /* si format court et 7 fichiers sur ligne */
   fprintf(stdout, "\n");
```

```
    if(forml)                          /*  si format long, affichage nb fichiers */
      {
       fprintf(stdout, "   %d files / %ld bytes \n", nb_files, ltotalo);
       fprintf(stdout, "   %ld bytes free\n", (2462L-(longd+ltotale))*256);
      }

    fin(0);                  /* réinitialisation du lecteur et fin du programe */
  }
```

```
/*****************************************************************************/
/*****************************************************************************/
/**                                LIFCP.C                                  **/
/**            Copie de fichiers TEXT (disquette LIF <-> MS/DOS)            **/
/**                                                                         **/
/**    Création : 20/10/90                Auteur : JB                       **/
/**    Derniere modification : 18/03/92                                     **/
/**        Modification des tables de conversion                            **/
/**  NOTE : Ne pas oublier de compiler ■ mode LARGE ou COMPACT              **/
/*****************************************************************************/
/*****************************************************************************/
#include "lif.c"
#include "getopt.c"

#if defined(__TURBOC__)        /* fonctions dépendant du type de compilateur */
#include "lif.tc"
#else
#include "lif.msc"
#endif

      /* table de conversion de caractères IBM vers Roman8 (à partir du 128e)*/
unsigned char ibm_r8[]={180, 207, 197, 192, 204, 200, 212, 181,   /* 128-135 */
                        193, 205, 201, 221, 209, 217, 216, 208,   /* 136-143 */
                        220, 215, 211, 194, 206, 202, 195, 203,   /* 144-151 */
                        239, 218, 219, 191, 187, 188, '?', 190,   /* 152-159 */
                        196, 213, 198, 199, 183, 182, 249, 250,   /* 160-167 */
                        185, '?', '?', 248, 247, 184, 251, 253,   /* 168-175 */
                        '?', '?', '?', '?', '?', '?', '?', '?',   /* 176-183 */
                        '?', '?', '?', '?', '?', '?', '?', '?',   /* 184-191 */
                        '?', '?', '?', '?', '?', '?', '?', '?',   /* 192-199 */
                        '?', '?', '?', '?', '?', '?', '?', '?',   /* 200-207 */
                        '?', '?', '?', '?', '?', '?', '?', '?',   /* 208-215 */
                        '?', '?', '?', '?', '?', '?', '?', '?',   /* 216-223 */
                        '?', 222, '?', '?', '?', '?', 243, '?',   /* 224-231 */
                        '?', '?', '?', '?', '?', 210, '?', '?',   /* 232-239 */
                        '?', 254, '?', '?', '?', '?', '?', '?',   /* 240-247 */
                        179, 242, 242, '?', '?', '?', 252, 160};  /* 248-255 */


     /* table de conversion de caractères Roman8 vers IBM (à partir du 128e) */
unsigned char r8_ibm[]={'?', '?', '?', '?', '?', '?', '?', '?',   /* 128-137 */
                        '?', '?', '?', '?', '?', '?', '?', '?',   /* 136-143 */
                        '?', '?', '?', '?', '?', '?', '?', '?',   /* 144-151 */
                        '?', '?', '?', '?', '?', '?', '?', '?',   /* 152-159 */
                        255, '?', '?', '?', '?', '?', '?', '?',   /* 160-167 */
                        '?', '?',  94, '?', '?', '?', '?', 156,   /* 168-175 */
                        '?', '?', '?', 248, 128, 135, 165, 164,   /* 176-183 */
                        179, 168, '?', 156, 157, '?', 159, 155,   /* 184-191 */
                        131, 136, 147, 150, 160, 130, 162, 163,   /* 192-199 */
                        133, 138, 149, 151, 132, 137, 148, 154,   /* 200-207 */
                        143, 140, 237, 146, 134, 161, 237, 145,   /* 208-215 */
                        142, 141, 153, 154, 144, 139, 225, '?',   /* 216-223 */
```

```
                        '?', '?', '?', '?', '?', '?', '?', '?',   /* 224-231 */
                        '?', '?', '?', '?', '?', '?', '?', 152,   /* 232-239 */
                        '?', '?', 249, 230, ' ', ' ', 196, 172,   /* 240-247 */
                        171, 166, 167, 174, 254, 175, 241, 255};  /* 248-255 */


void main(int argc, char *argv[])
  {
   int    sens=0;                        /* 1 : LIF->MS/DOS   2 : MS/DOS->LIF */
   int    mode_bin=0;              /* traitement des fichiers [redacted] mode binaire */
   int    type_in=0, type_out=0; /* types des fichiers 1 : LIF   0 : MS-DOS */
   int    lif_drive;           /* numéro du lecteur de disquette (0=A, 1=B) */
   int    err_entry=0;                /* flag d'erreur entrée de parametres */
   FILE   *fp;                          /* descripteur du fichier MS/DOS */
   LIFDIR dir;                 /* structure contenant l'entrée du directory */
   int    pos_dir;                      /* numéro de l'entrée du directory */
   char   lif_name[11], dos_name[50];                    /* [redacted] des fichiers */
   int    type;                                /* code du type de fichier */
   int    private, secure; /* codes de sécurité et privatisation du fichier */
   GDH    gdh;                  /* structure contenant la date du fichier */
   long   imp1, imp2; /*codes specifiques aux types de fich. dans directory */
   int    long_file;          /* longueur du fichier LIF [redacted] enregistrements */
   int    first_sect;            /* premier enregistrement du fichier LIF */
   int    cl_dir;             /* si déplacement de fichier, ancienne entrée */
   int    nb_char;                      /* nombre de caractères de la chaine */
   long   nb_chars;
   unsigned char far *dos_string;      /*chaine de caractères traitée (DOS) */
   unsigned char out_char;                                /* caractère traité */
   unsigned char c_fort;              /* poids fort de la longueur de chaine */
   unsigned char c_faible;          /* poids faible de la longueur de chaine */
   int    retour, retour2;    /*codes de retour des fct de lecture/ecriture */
   long   chk_file;              /* variable de controle de fichier LIF/DOS */
   char   garbage[50];
   int    i;
   char   c;

   SW='-';
   while((c=(char) getopt(argc, argv, "b")) != EOF)
     {
      switch(c)
        {
         case 'b' : mode_bin = 1;                    /* copie en mode binaire */
                    break;

         case '?' : err_entry++;                                /* si erreur */
                    break;
        }
     }

   if(!err_entry)                                        /* si pas d'erreur */
     {
      if((argc-optind) == 2)             /* 2 noms de fichiers opbligatoires */
        {
                                                  /* détection du fichier LIF */
         if((argv[optind][0] >= '0') && (argv[optind][0] <= '1'))
            type_in=1;
         if((argv[optind+1][0] >= '0') [redacted] (argv[optind+1][0] <= '1'))
            type_out=1;
         if((type_in+type_out) != 1)          /* 1 seul fichier LIF autorisé */
            err_entry=1;
        }
      else
        {
         err_entry = 1;
        }
```

```
     }

   if(err_entry)                                                /* si erreur */
     {
      fprintf(stderr, "usage : lifcp [-b] drive_number:LIF_FILE dos_file\n");
      fprintf(stderr, "    or  lifcp [-b] dos_file drive_number:LIF_FILE\n");
      exit(-1);
     }

   if(type_in==1)                          /* determination du [redacted] de copie */
     {
      sens=1;                                                  /* [redacted] LIF vers MS/DOS */
      lif_drive=(int) argv[optind][0]-'0';   /* selection numéro de lecteur */
      strcpy(lif_name, argv[optind]+2);   /* acquisition nom de fichier LIF */
      strcpy(dos_name, argv[optind+1]);   /* acquisition nom de fichier DOS */
     }
   else
     {
      sens=2;
      lif_drive=(int) argv[optind+1][0]-'0';
      strcpy(lif_name, argv[optind+1]+2);
      strcpy(dos_name, argv[optind]);
     }

   for(i=strlen(lif_name); i<10; i++)     /* ajout d' espaces en fin de nom */
      lif_name[i]=' ';
   lif_name[10]='\0';

   init_dsk();                     /* initialisation de lecteur de disquettes */

   if(sens==1)                                     /* si copie LIF vers MS/DOS */
     {
                                  /* recherche du fichier sur la disquette LIF */
      retour=sch_dir(lif_drive, lif_name, &pos_dir, &dir);
      if(retour < 0)
        {
         fprintf(stderr, "file %s not found\n", lif_name);
         fin(-1);
        }
                                   /* décomposition de l'entrée de directory */
      break_dir(lif_name, &private, &secure, &type, &first_sect, &long_file,
                                        &gdh, &imp1, &imp2, &dir);

      if(mode_bin)                                 /* si copie en mode binaire */
        {
         fp=fopen(dos_name, "wb");   /* ouverture (ou création) fichier DOS */

         chk_file=123123L;              /* écriture de la valeur de controle */
         retour=fwrite(&chk_file, sizeof(long), 1, fp);
         if(retour != 1)                              /* si erreur d'écriture */
           {
            fprintf(stderr, "Error : can't write %sn\n", dos_name);
            fin(-1);
           }

         retour=fwrite(&dir, 32, 1, fp);   /* écriture entrée de répertoire */

         i=0;          /* copie des enregistrements LIF dans le fichier DOS */
         do
           {
            curr_sect=first_sect+i;
            retour2=read_dsk(lif_drive, curr_sect, (char *) buff_sect);
            if(retour2 == 0)
               retour=fwrite(buff_sect, 256, 1, fp);
            i++;
           }
```

```
        while((i != long_file) && (retour == 1) && (retour2 == 0));

        if(retour2 != 0)        /* si erreur de lecture de la disquette LIF */
          {
           fprintf(stderr, "I can't read the file %s\n", lif_name);
           fin(-1);
          }

        if(retour != 1)         /* si erreur d'ecriture dans le fichier DOS */
          {
           fprintf(stderr, "I can't write the file %s\n", dos_name);
           fin(-1);
          }
      }
    else                                        /* si copie en mode TEXT */
      {
        if(type != 1)                     /* si ce n'est pas un fichier TEXT */
          {
           fprintf(stderr, "%s: Not a TEXT file\n", lif_name);
           fin(-1);
          }

        fp=fopen(dos_name, "w");    /* ouverture (ou création) fichier DOS */


        dos_string=farmalloc(MAX_BUFFSIZE); /*allocation mémoire pr chaine */

        curr_char=255;        /* initialisation position pointeur caractere */
        curr_sect=first_sect;
                                  /* lecture séquentielle du fichier LIF */
        while(((c_fort=read_char(lif_drive)) != 255) &&
              ((c_faible=read_char(lif_drive)) != 255))
          {
           nb_char=((int)c_fort<<8)+c_faible; /*nbre de caract. de la ligne*/
           for(i=0; i<nb_char; i++)                  /* lecture de la ligne */
             {
              if((out_char=read_char(lif_drive)) > 127) /*conversion R8-IBM*/
                 out_char=r8_ibm[out_char-128];
              dos_string[i]=out_char;
             }
           dos_string[nb_char]='\0';
                                          /* écriture ligne dans fichier DOS*/
           fprintf(fp, "%s\n", (char far *) dos_string);

           if(nb_char & 0x0001)  /* si nombre impair de caract. dans ligne */
              read_char(lif_drive);                /* ..on saute un caractere */
          }
        farfree(dos_string);                            /* liberation mémoire */
      }
    fclose(fp);                                       /* fermeture fichier DOS */

    set_dos_gdh(dos_name, &gdh);                      /* maj date du ficher dos */
  }
else                                                  /* si copie MS/DOS -> LIF */
  {
   if(mode_bin)                                      /* si copie en mode binaire */
     {
      fp=fopen(dos_name, "rb");  /*ouverture fich. DOS en lecture+binaire*/
      if(fp == NULL)
        {
         fprintf(stderr, "file %s not found\n", dos_name);
         fin(-1);
        }

                  /* lecture de la valeur de controle de fichier LIF/DOS */
      retour=fread(&chk_file, sizeof(long), 1, fp);
      if(retour != 1)
        {
         fprintf(stderr, "I can't read %s\n", dos_name);
         fin(-1);
        }

                                  /* si la valeur lue n'est pas le code attendu */
      if(chk_file != 123123L)
        {
         fprintf(stderr, "%s is not a LIF/DOS file\n", dos_name);
         fin(-1);
        }
                                          /* lecture de l'entrée de répertoire */
      retour=fread(&dir, 32, 1, fp);
      if(retour != 1)
        {
         fprintf(stderr, "I can't read %s\n", dos_name);
         fin(-1);
        }

      /* récuperation de la longueur en enregistrements dans 'long_file' */
      break_dir(garbage, &private, &secure, &type, &first_sect, &long_file,
                                  &gdh, &imp1, &imp2, &dir);

              /* recherche d'une place sur la disquette pour le fichier */
      first_sect=best_room(lif_drive, lif_name, long_file, &pos_dir,
                                                          &cl_dir);
      if(first_sect == -3)                    /* si disquette non reconnue */
        {
         fprintf(stderr, "Not a LIF disk\n");
         fin(-1);
        }
      if(first_sect == -1)                              /* si plus de place */
        {
         fprintf(stderr, "Not enough space for this file\n");
         fin(-1);
        }

      i=0; /* copie du fichier sur la disquette LIF (par enregistrement) */
      do
        {
         curr_sect=first_sect+i;
         retour=fread(buff_sect, 256, 1, fp);
         if(retour == 1)
            retour2=wrt_dsk(lif_drive, curr_sect, (char *) buff_sect);
         i++;
        }
      while((i != long_file) && (retour == 1) && (retour2 == 0));

      if(retour != 1)              /* si erreur de lecture du fichier DOS */
        {
         fprintf(stderr, "I can't read the file %s\n", dos_name);
         fin(-1);
        }

      if(retour2 != 0)       /* si erreur d'écriture sur la disquette LIF */
        {
         fprintf(stderr, "I can't write the file %s\n", lif_name);
         fin(-1);
        }

      memcpy(&dir, lif_name, 10);     /* copie du parametre comme nom LIF */
     }
   else                                                /* si copie en mode TEXT */
     {
      fp=fopen(dos_name, "r");        /* ouverture fichier DOS en lecture */
```

```
if(fp == NULL)
  {
   fprintf(stderr, "file %s not found\n", dos_name);
   fin(-1);
  }
dos_string=farmalloc(MAX_BUFFSIZE); /*allocation mémoire pr chaine */

                                  /* lecture totale du fichier DOS
                              afin d'en déterminer la longueur exacte */
i=0;
nb_chars=0L;
while(fgets((char far *) dos_string, 32767, fp) != NULL)
  {
   nb_char = strlen((char far *) dos_string);
   if(dos_string[nb_char-1] == 0x0A)
     {
      nb_char--;
     }
   nb_chars += (long) nb_char+2;
   /* prise en compte caractère suplémentaire si nb caract. impair */
   if(nb_char & 0x00000001)
      nb_chars++;
  }
nb_chars += 2L;

if(nb_chars & 256L)              /* longueur fichier LIF en enreg. */
   long_file=(int) ((nb_chars/256L)+1);
else
   long_file=(int) (nb_chars/256L);

fclose(fp);                              /* fermeture du fichier DOS */

        /* recherche d'une place sur la disquette pour le fichier */
first_sect=best_room(lif_drive, lif_name, long_file, &pos_dir,
                                                        &cl_dir);
if(first_sect == -3)                  /* si disquette non reconnue */
  {
   fprintf(stderr, "Not a LIF disk\n");
   fin(-1);
  }
if(first_sect == -1)                            /* si plus de place */
  {
   fprintf(stderr, "Not enough space for this file\n");
   fin(-1);
  }

get_dos_gdh(dos_name, &gdh);    /* acquisition date du fichier DOS */

fp=fopen(dos_name, "r");    /* ouverture du fichier DOS en lecture */

curr_sect=first_sect;
curr_char=0;
while(fgets((char far *) dos_string, 32767, fp) != NULL) /*lecture */
  {
   nb_char=strlen((char far *) dos_string); /*nb caract. par ligne */
   if(dos_string[nb_char-1] == 0x0A)
     {
      nb_char--;
      dos_string[nb_char] = '\0';
     }
                              /* ecriture de la longueur de la ligne */
   write_char(lif_drive, (unsigned char) nb_char / 256);
   write_char(lif_drive, (unsigned char) nb_char & 256);
   for(i=0; i<nb_char; i++)              /* ecriture des caractères */
     {
      out_char=dos_string[i];
      if(out_char > 127)            /* conversion caract IBM -> R8 */
         out_char = ibm_r8[out_char-128];
      write_char(lif_drive, out_char);     /* ecriture dans buffer */
     }
   if(nb_char & 0x0001)           /* si nombre impair de caractères */
      write_char(lif_drive, 'A');     /* on ajoute 1 quelconque */
  }

farfree(dos_string);                          /* libération mémoire */
                                   /* réinitialisation fin du buffer */
memset(buff_sect+curr_char, 0xFF, 256-curr_char);
                                  /* ecriture du contenu courant du buffer */
wrt_dsk(lif_drive, curr_sect, (char *) buff_sect);

                                  /* construction de l'entrée du directory */
build_dir(&dir, lif_name, 0, 0, TEXT, first_sect, long_file,
                                                        &gdh, 0L, 0L);
 }
wrt_dir(lif_drive, pos_dir, &dir);            /* ecriture du directory */

if(cl_dir != -1)           /* si fichier 'déplacé' sur la disquette... */
   clr_file(lif_drive, cl_dir);     /* ...on efface l'ancienne entrée */

fclose(fp);                                /* fermeture du fichier DOS */
 }
fin(0);                /* réinitialisation du lecteur et fin du programme */
}
```

```
/*****************************************************************************/
/*****************************************************************************/
/**                                ACP.C                                    **/
/**    Conversion des fichiers générés par AREUH & LEX sur disquette LIF   **/
/**                                                                         **/
/**    Création : 30/10/90              Auteur : JB (basé sur COPY.H       **/
/**                                                   de pda et JT)         **/
/**    Derniere modification : 16/02/91                                     **/
/**       Ajout du paramètre LIF_NAME                                       **/
/**                                                                         **/
/*****************************************************************************/
/*****************************************************************************/
#include "lif.c"

#if defined(__TURBOC__)
#include "lif.tc"
#else
#include "lif.msc"
#endif

long read_nib(FILE *fp, int c);         /* prototype des fonctions utilisées */
void skip(FILE *fp, int n);
int  hex(unsigned char c);

void main(int argc, char *argv[])
  {
   int    lif_drive=0;          /* numéro du lecteur de disquette (0=A, 1=B) */
   int    opt_file=0;   /* flag de présence de nom de fichier LIF optionnel */
   int    err_entry=0;                  /* flag d'erreur entrée de parametres */
   FILE   *fp;                                /* descripteur du fichier MS/DOS */
   LIFDIR dir;                   /* structure contenant l'entrée du directory */
   int    pos_dir;                          /* numéro de l'entrée du directory */
   char   lif_name[11], dos_name[50];                        /* nom des fichiers */
   GDH    gdh;                        /* structure contenant la date du fichier */
```

```
int     long_file;          /* longueur du fichier LIF en enregistrements */
int     first_sect;             /* premier enregistrement du fichier LIF */
int     cl_dir;             /* si déplacement de fichier, ancienne entrée */
long    nb_bytes;                           /* nombres d'octets du LEX */
unsigned long magic;                    /* code de validité du fichier */
unsigned long lnib;                     /* longueur fichier en nibbles */
int     i;

if((argc<2) || (argc>3))                /* 1 ou 2 paramètres obligatoires */
  {
   err_entry=1;
  }
else
  {
   if(argc==3)                                          /* si 2 paramètres */
     {
            /* la longueur doit être comprise entre 2 et 12 caractères */
      if((strlen(argv[2]) < 2) || (strlen(argv[2]) > 12))
         err_entry = 1;
      lif_drive=(int) argv[2][0]-'0';       /* controle numéro de lecteur */
      if((lif_drive < 0) || (lif_drive > 1))
         err_entry=1;
      if(!err_entry)
        {
         if(argv[2][1] != ':')              /* controle deuxième caractère */
            err_entry=1;
         if(!err_entry && (strlen(argv[2]) > 2))  /* acquisition nom LIF */
           {
            opt_file=1; /* armement du flag de présence de nom optionnel */
            strcpy(lif_name, argv[2]+2);
                                            /* ajout d'espaces en fin du nom */
            for(i=strlen(lif_name); i<10; i++)
               lif_name[i]=' ';
            lif_name[10]='\0';
           }
        }
     }
  }

if(err_entry)                                           /* si erreur */
  {
   fprintf(stderr, "usage : acp file [drive_number:[LIF_NAME]]\n");
   exit(-1);
  }

strcpy(dos_name, argv[1]);              /* récupération nom du fichier DOS */

init_dsk();                                  /* initialisation du lecteur */

fp=fopen(dos_name, "rb");                     /* ouverture du fichier DOS */
if(fp == NULL)                                          /* si non trouvé...*/
  {
   fprintf(stdout, "file %s not found\n", dos_name);
   fin(-1);
  }

                          /* lecture de la valeur de controle du fichier */
fread ((char *) &magic, sizeof (unsigned long), 1 , fp);
if (magic!=0x1b080100L)                                 /* si code different ... */
  {
   fprintf(stdout, "Error : %s is not a LEX file\n", dos_name);
   fin(-1);
  }

if(opt_file)                /* si présence du nom de fichier en paramètre */
  {
    skip(fp, 32);          /* on saute le nom, le type, la date et l'heure */
  }
else
  {
   for(i=0; i<8; i++)                /* lecture et formatage du nom du LEX */
      lif_name[i] = (unsigned char) read_nib(fp, 2);
   lif_name[8] = lif_name[9] = ' ';
   lif_name[10] = '\0';
   skip(fp, 16);       /* on saute le type, la date et l'heure du fichier */
  }

                          /* lecture de la longueur en nibbles du fichier LEX */
lnib = ((unsigned long) read_nib(fp, 5)) - (unsigned long) 5;
                                        /* calcul de la longueur en enregistrements */
long_file = (int) (lnib / 512) + ((lnib % 512) ? 1 : 0) ;


                  /* recherche d'une place sur la disquette pour la fichier */
first_sect=best_room(lif_drive, lif_name, long_file, &pos_dir, &cl_dir);
if(first_sect == -3)
  {
   fprintf(stdout, "Error : Not a LIF disk\n");
   fin(-1);
  }
if(first_sect == -1)
  {
   fprintf(stdout, "Not enough space for this file\n");
   fin(-1);
  }

curr_sect=first_sect; /* initialisations des pointeurs pour 'write char' */
curr_char=0;
nb_bytes = ((lnib % 2) ? lnib+1 : lnib) >> 1;  /* longueur LEX en octets */
while(nb_bytes--)
  {                                       /* lecture et conversion du fichier */
   write_char(lif_drive, (unsigned char) read_nib(fp, 2));
  }
memset(buff_sect+curr_char, 0x00, 256-curr_char); /*normalisation fin enr*/
wrt_dsk(lif_drive, curr_sect, (char *) buff_sect); /*écriture dernier enr*/

fclose(fp);                                     /* fermeture du fichier DOS */

get_dos_gdh(dos_name, &gdh);      /* lecture date et heure du fichier DOS */

                                        /* construction entrée du directory */
build_dir(&dir, lif_name, 0, 0, LEX, first_sect, long_file, &gdh,lnib, 0L);
wrt_dir(lif_drive, pos_dir, &dir);        /* ecriture entrée du directory */

if(cl_dir != -1)                                        /* si fichier déplacé */
   clr_file(lif_drive, cl_dir);            /* on efface l'ancienne entrée */

fin(0);                /* fin du programme et réinitialisation du lecteur */
 }


/*****************************************************************************/
/*   READ_NIB : lit, inverse et convertit 'n' nibbles du fichier             */
/*   entrées : fp = descripteur de fichier DOS                               */
/*             n = nombre de nibbles à lire                                  */
/*   valeur retournée : nibbles lus, convertis en décimal                    */
/*****************************************************************************/
long read_nib(FILE *fp, int n)
  {
   unsigned char buf[16];              /* buffer d'inversion des caractères */
   int i;
   long res;                                            /* valeur calculée */
```

```
    res = 0;
    for (i=0; i<n; i++)
      {                  /* lecture de 'n' caractères (nibbles) dans le buffer */
       buf[i] = (unsigned char) getc(fp);
      }
    for(i=n-1; i>=0; i--)                  /* lecture buffer en ordre inverse */
       res = (res << 4) | hex (buf [i]);  /* calcul res = res*16+hex(buf[i]) */
    return (res);
   }

/*****************************************************************************/
/*   SKIP : Saute d'un nombre 'n' de caractères dans le fichier              */
/*   entrées : fp = descripteur de fichier DOS                               */
/*             n = nombre de nibbles à sauter                                */
/*   valeur retournée : aucune                                               */
/*****************************************************************************/
void skip(FILE *fp, int n)
   {
    unsigned char buf[32];                          /* buffer de réception */

    fread(buf, n, 1, fp);                          /* lecture de 'n' nibbles */
   }

/*****************************************************************************/
/*   HEX : convertit un caractere hexadecimal en nombre décimal              */
/*   entrée : c = caractère à convertir                                      */
/*   valeur retournée : nombre décimal                                       */
/*****************************************************************************/
int hex(unsigned char c)
   {
    return (((c>='A')&&(c<='F')) ? c-'A'+10 : c-'0') ;
   }
```

```
/*****************************************************************************/
/*****************************************************************************/
/**                              GETOPT.C                                   **/
/**       Fonction d'analyse des paramètres des commandes (Style UNIX)      **/
/**                                                                         **/
/**   Dernière modification : 07/02/91                                      **/
/**     Ajout de la selection du caractère d'indication d'option par 'SW'   **/
/**                                                                         **/
/*****************************************************************************/
/*****************************************************************************/
/*                                                                           */
/*                              Description                                  */
/*                                                                           */
/*  Cette fonction permet d'analyser les paramètres optionnels               */
/*  d'une commande MS/DOS dans le style utilisé par UNIX.                    */
/*                                                                           */
/*      commande [-options] [opérandes...]                                   */
/*                                                                           */
/*   par exemple la commande :                                               */
/*                                                                           */
/*       dircat  [-d] [-s] [-w width]  file1 [file2...]                      */
/*                                                                           */
/*   comporte 3 options : d, s, et w. (cette derniere option comporte en     */
/*     outre un argument). Ces options peuvent être tapées dans n'importe    */
/*     quel ordre ou être groupées :                                         */
/*                                                                           */
/*       dircat  -s -w 15 -d  toto.txt                                       */
/*       dircat  -ds  toto.txt titi.txt                                      */
/*                                                                           */
/*  UNIX System V admet les règles suivantes pour la syntaxe des commandes:  */
/*                                                                           */
/*  Règle 1  : La longueur des noms de commandes doit être comprise entre    */
/*             2 et 9 caractères.     - non valable pour MS/DOS (1 et 8) -  */
/*  Règle 2  : Les noms de commandes ne doivent inclure que des lettres      */
/*             minuscules et des chiffres.     - non valable pour MS/DOS -   */
/*  Règle 3  : Les noms d'option doivent avoir une longueur d'un caractère   */
/*  Règle 4  : Toutes les options doivent commencer par le délimiteur '-'    */
/*  Règle 5  : Les options sans arguments peuvent être groupées derrière     */
/*             le même délimiteur.                                           */
/*  Règle 6  : Le premier argument d'option suivant une option doit être     */
/*             précédée par un espace.                                       */
/*  Règle 7  : Les arguments d'option ne peuvent pas être optionnels          */
/*  Règle 8  : Les groupes d'arguments d'option suivant une option           */
/*             doivent être séparés par des virgules, ou séparés par des     */
/*             espaces et entourés par des apostrophes.                      */
/*  Règle 9  : Toutes les options doivent précéder les opérandes dans la     */
/*             ligne de commande.                                            */
/*  Règle 10 : Les caractères '--' peuvent être utilisés pour délimiter      */
/*             la fin des options.                                           */
/*  Règle 11 : L'ordre des options est indiférent.                           */
/*  Règle 12 : L'ordre des opérandes et leur interprétation doit être        */
/*             déterminé par une analyse spécifique dans la commande.        */
/*  Règle 13 : Le caractère '-' précédé et suivi par un espace doit          */
/*             être utilisé seulement pour spécifier l'entrée standard.      */
/*                                                                           */
/*  La fonction 'getopt', incluse dans l'analyse des paramètres de la        */
/*  commande, permet de tenir compte de ces règles et de simplifier          */
/*  cette analyse.                                                           */
/*                                                                           */
/*  'getopt' retourne la lettre d'option suivante, contenue dans 'argv',     */
/*  qui correspond à 'optstring'. Quand toutes les options ont été           */
/*  décodées, la fonction retourne EOF. Si une option inconnue est           */
/*  rencontrée, la fonction retourne '?'.                                    */
/*                                                                           */
/*  'optarg' contient (si l'option le permet) l'argument correspondant à     */
/*  l'option rencontrée.                                                     */
/*                                                                           */
/*  'optind' contient l'index de 'argv' correspondant à la prochaine         */
/*  option à décoder, ou à la première opérande aprés la fin de l'analyse    */
/*  des options.                                                             */
/*                                                                           */
/*  'optstring' est une chaine de caractères contenant les lettres des       */
/*  options admises. Si une lettre est suivie de ':', l'option doit          */
/*  comporter un argument.                                                   */
/*                                                                           */
/*  Exemple :                                                                */
/*                                                                           */
/*  Analyse de la commande :  comm [-a] [-b] [-o file] extfile...            */
/*                                                                           */
/*     #include "getopt.c"                                                   */
/*                                                                           */
/*     main(int argc, char *argv[])                                          */
/*         {                                                                 */
/*          int opta, optb, opto;                                            */
/*          char ofile[40];                                                  */
/*          char c;                                                          */
/*          int errflag;                                                     */
/*                                                                           */
/*          opta=0;                      / initialisation des options à 0 /  */
/*          optb=0;                                                          */
/*          opto=0;                                                          */
/*                                                                           */
/*          while((c=getopt(argc, argv, "ao:b")) != EOF)                     */
/*            {                                                              */
```

```
/*              switch(c)                          / selection des options /  */
/*                {                                                            */
/*                 case 'a' : opta = 1;                                        */
/*                            break;                                           */
/*                                                                             */
/*                 case 'b' : optb = 1;                                        */
/*                            break;                                           */
/*                                                                             */
/*                 case 'o' : opto = 1;                                        */
/*                            strcpy(ofile, optarg);                           */
/*                            break;                                           */
/*                                                                             */
/*                 case '?' : errflag++;                                       */
/*                            break;                                           */
/*                }                                                            */
/*              }                                                              */
/*                                                                             */
/*            if(errflag)                          / sortie en cas d'erreur /  */
/*              {                                                              */
/*               fprintf(stderr, "usage : comm [-a] [-b] [-o file] extfile"); */
/*               exit(-1);                                                     */
/*              }                                                              */
/*                                                                             */
/*            for(; optind<argc; optind++)         / analyse des opérandes /   */
/*              {                                                              */
/*               if(access(argv[optind], 4)                                    */
/*                 {                                                           */
/*                 .                                                           */
/*                 .                                                           */
/*                 .                                                           */
/*            }                                                                */
/*                                                                             */
/*   Divergences de la fonction incluse dans ce listing :                      */
/*                                                                             */
/*     1 - Il est possible de changer le caractère délimiteur '-' par          */
/*         un autre caractère (par exemple '/', plus utilisé par MS/DOS)       */
/*         en placant ce caractère dans la variable 'SW' avant l'appel         */
/*         de 'getopt'.                                                        */
/*     2 - La règle 13 n'est pas implantée.                                    */
/*     3 - Les versions actuelles d'UNIX ne satisfont pas aux règles 5 et 6.   */
/*         En effet il est actuellement possible de taper :                    */
/*                              cmd -abo xxx                                   */
/*                au lieu de    cmd -ab -o xxx      (violation règle 5)          */
/*         ou                   cmd -ab -oxxx                                  */
/*                au lieu de    cmd -ab -o xxx      (violation règle 6)          */
/*         Cette facon d'entrer les paramètres pouvant ne plus être supporté   */
/*         dans une prochaine version d'UNIX. Cependant, afin de ne pas        */
/*         changer les (mauvaises !) habitudes, ce type de syntaxe à           */
/*         été implanté dans ce listing.                                       */
/*                                                                             */
/*                                                                             */
#include <string.h>
#include <stdio.h>

char *optarg;
int optind=1;
int SW = '-';

int getopt(int argc, char *argv[], char *optstring)
   {
    char *ptopt;
    int retour='?';
    int len_arg;

    optarg=NULL;

    if(optind == argc)                        /* si fin de la ligne de commande */
       return(EOF);

    if(*argv[optind] == SW)                   /* si parametre commence par '-' */
      {
       len_arg=strlen(argv[optind]);                /* calcul longueur parametre */
       retour=argv[optind][1];            /* acquisition du caractère d'option */
       if(retour == SW)                                        /* si option '--' */
         {
          optind++;                              /* selection parametre suivant */
          return(EOF);                            /* fin analyse des paramètres */
         }
       ptopt=strchr(optstring, retour);   /* recherche option dans optstring */
       if(ptopt == NULL)                              /* si option non trouvée */
         {
          optind++;
          return('?');                                  /* sortie avec erreur */
         }
       if(*(ptopt+1) == ':')          /* si option suivi de ':' dans optstring */
         {
          if(len_arg == 2)          /* si option entrée dans la forme '-o xxx' */
            {
             optind++;                                   /* acces à l'argument */
             optarg=argv[optind];                       /* acquisition argument */
                                          /* si option non suivie d'un argument */
             if((optind>=argc) || (*optarg=='-'))
               {
                optind++;
                return('?');                             /* sortie avec erreur */
               }
            }
          else                        /* si option entrée sous la forme '-oxxx' */
            {
             optarg=argv[optind]+2;                     /* acquisition argument */
            }
         }
       else          /* si caractere option non suivie de ':' dans optstring */

         {
          if(len_arg != 2)                     /* si options groupées ('-abc') */
            {
             argv[optind]++;             /* préparation du passage suivant ... */
             *argv[optind]=SW;          /* ... en transformant '-abc' en '-bc' */
             optind--;
            }
         }
      }
    else                                /* si le paramètre n'est pas une option */
      {
       return(EOF);                          /* fin de l'analyse des paramètres */
      }
    optind++;                                   /* selection parametre suivant */
    return (retour);                           /* retour du caractère d'option */
   }
```

Le Journal JPC est le bulletin de liaison entre les membres de l'Association "PPC Paris", régie par la loi de 1901. Le Club est éditeur de JPC, et son siège social est au 56, rue Jean-Jacques Rousseau, 75001 Paris.

La maquette de ce numéro a été préparée et réalisée par Jacques Belin et Asdin Aoufi.

Les dessins sont de Jean-Jacques Dhénin et Paul Courbis.

Directeur de la publication : Jacques Belin
Numéro ISSN : 0762 - 381X

Veuillez adresser toute correspondance à :
PPC Paris, BP 604, 75028 Paris Cedex 01.